Single Root I/O Virtualization and Sharing
Revision 0.7

January 11, 2007

REVISION	REVISION HISTORY	DATE
0.5	Initial release	07/24/2006
0.7	Methods defined	1/11/2007

PCI-SIG disclaims all warranties and liability for the use of this document and the information contained herein and assumes no responsibility for any errors that may appear in this document, nor does PCI-SIG make a commitment to update the information contained herein.

Contact the PCI-SIG office to obtain the latest revision of the specification.

Questions regarding this document or membership in PCI-SIG may be forwarded to:

Membership Services
http://www.pcisig.com
E-mail: administration@pcisig.com
Phone: 503-291-2569
Fax: 503-297-1090

Technical Support
Technical support for this specification is available to members. For information, please visit: http://www.pcisig.com/developers/technical_support.

DISCLAIMER

This document is provided "as is" with no warranties whatsoever, including any warranty of merchantability, non-infringement, fitness for any particular purpose, or any warranty otherwise arising out of any proposal, specification, or sample. PCI-SIG disclaims all liability for infringement of proprietary rights, relating to use of information in this specification. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted herein.

PCI Express, PCIe, PCI-X, and PCI-SIG are trademarks of PCI-SIG.

All other product names are trademarks, registered trademarks, or service marks of their respective owners.

Copyright © 2007 PCI-SIG
All rights reserved.

Contents

1. ARCHITECTURAL OVERVIEW 13

1.1.1. Virtualization Technology Overview 13
1.1.2. PCI Technologies 14
1.1.3. Interoperability Requirements 14
1.1.4. Related Technologies Outside of Scope 15
1.2. EXAMPLE USAGE MODELS 15
1.3. SINGLE ROOT (SR) ARCHITECTURAL ELEMENTS 16
1.4. EXAMPLE OPERATIONAL FLOWS 17
1.5. TBD 17

2. INITIALIZATION AND RESOURCE ALLOCATION 18

2.1. SR-IOV RESOURCE DISCOVERY 18
2.1.1. SR-IOV Capability Discovery 18
2.1.2. Configuring SR-IOV Capabilities 19
2.1.2.1. Configuring the BAR Mechanisms 19
2.1.3. VF Discovery 20
2.1.4. Function Dependency Lists 22
2.1.5. Interrupt Resource Allocation 22
2.1.6. Resources Not Supported List 22
2.2. RESET MECHANISMS 22
2.2.1. Conventional Reset 23
2.2.2. FLR that targets a VF 23
2.2.3. FLR that targets a PF 23
2.3. IOV RE-INITIALIZATION AND REALLOCATION 23
2.4. VF MIGRATION 24

3. CONFIGURATION 27

3.1. REQUIREMENTS 27
3.1.1. PF Requirements 27
3.1.2. VF Requirements 27
3.1.3. Virtual Intermediary (VI) Requirements 27
3.1.4. SR-PCIM Requirements 28
3.2. CONFIGURATION SPACE 29
3.3. SR-IOV EXTENDED CAPABILITY 29
3.3.1. SR-IOV Extended Capability Header (00h) 29
3.3.2. SR-IOV Capabilities (04h) 30
3.3.2.1. VF Migration Capable 30
3.3.2.2. VF Migration Interrupt Message Number 31
3.3.3. SR-IOV Control (08h) 31
3.3.3.1. VF Enable 31
3.3.3.2. VF Migration Enable 32
3.3.3.3. VF Migration Interrupt Enable 32
3.3.3.4. VF MSE (Memory Space Enable) 32

3.3.3.5. VF ARI Enable 32
3.3.4. SR-IOV Status (0Ah) *33*
3.3.4.1. VF Migration Event Pending 33
3.3.5. InitialVFs (0Ch) *33*
3.3.6. TotalVFs (0Eh) *33*
3.3.7. NumVFs (10h) *33*
3.3.8. Function Dependency Link (12h) *34*
3.3.9. First VF Offset (14h) *36*
3.3.10. VF Stride (16h) *36*
3.3.11. Supported Page Sizes (18h) *36*
3.3.12. System Page Size (1Ch) *36*
3.3.13. VF BAR0, VF BAR1, ... VF BAR5 (20h .. 34h) *37*
3.3.14. VF Migration State Array Offset (38h) *37*
3.3.14.1. VF Migration State Array 38
3.4. PF/VF CONFIGURATION SPACE HEADER 40
3.4.1. Type 0 Configuration Space Header *41*
3.4.1.1. Vendor ID Register (Offset 00h) 42
3.4.1.2. Device ID Register (Offset 02h) 42
3.4.1.3. Command Register (Offset 04h) 43
3.4.1.4. Status Register (Offset 06h) 44
3.4.1.5. Revision ID Register (Offset 08h) 44
3.4.1.6. Class Code (Offset 09h) 45
3.4.1.7. Cache Line Size Register (Offset 0Ch) 45
3.4.1.8. Latency Timer Register (Offset 0Dh) 45
3.4.1.9. Header Type (Offset 0Eh) 45
3.4.1.10. BIST (Offset 0Fh) 45
3.4.1.11. Base Address Registers (Offset 10h, 14h, … 24h) 45
3.4.1.12. Cardbus CIS Pointer (Offset 28h) 45
3.4.1.13. Subsystem Vendor ID (Offset 2Ch) 46
3.4.1.14. Subsystem Device ID (Offset 2Eh) 46
3.4.1.15. Expansion ROM BAR (Offset 30h) 46
3.4.1.16. Capabilities Pointer (Offset 34h) 46
3.4.1.17. Interrupt Line Register (Offset 3Ch) 46
3.4.1.18. Interrupt Pin Register (Offset 3Dh) 46
3.4.1.19. Min_Gnt / Max_Lat Registers (Offset 3Eh/3Fh) 46
3.5. PCI EXPRESS CAPABILITY 46
3.5.1. PCI Express Capability List Register (Offset 00h) *47*
3.5.2. PCI Express Capabilities Register (Offset 02h) *48*
3.5.3. Device Capabilities Register (Offset 04h) *48*
3.5.4. Device Control Register (Offset 08h) *50*
3.5.5. Device Status Register (Offset 0Ah) *51*
3.5.6. Link Capabilities Register (Offset 0Ch) *52*
3.5.7. Link Control Register (Offset 10h) *53*
3.5.8. Link Status Register (Offset 12h) *55*
3.5.9. Device Capabilities 2 Register (Offset 24h) *56*
3.5.10. Device Control 2 Register (Offset 28h) *56*

3.5.11. Device Status 2 Register (Offset 2Ah) *56*
3.5.12. Link Capabilities 2 Register (Offset 2Ch) *56*
3.5.13. Link Control 2 Register (Offset 30h) *56*
3.5.14. Link Status 2 Register (Offset 32h) *57*
3.6. PCI STANDARD CAPABILITIES 57
3.6.1. PCI Power Management Interface *57*
3.6.2. AGP Capability *57*
3.6.3. VPD Capability *57*
3.6.4. Slot Identification *58*
3.6.5. MSI *58*
3.6.6. CompactPCI Hot Swap *58*
3.6.7. PCI-X *58*
3.6.8. "Reserved for AMD" *58*
3.6.9. Vendor Specific *58*
3.6.10. Debug Port *58*
3.6.11. Compact PCI Central Resource Control *58*
3.6.12. PCI Hot Plug *58*
3.6.13. PCI Bridge Subsystem ID *58*
3.6.14. AGP 8x *59*
3.6.15. Secure Device (AMD) *59*
3.6.16. PCI Express *59*
3.6.17. MSI-X *59*
3.6.18. Serial ATA Data/Index Configuration *59*
3.7. PCI EXPRESS EXTENDED CAPABILITIES 59
3.7.1. Advanced Error Reporting *59*
3.7.2. Virtual Channel (02h) *59*
3.7.3. Device Serial Number *59*
3.7.4. Power Budgeting *59*
3.7.5. Root Complex Link Declaration *60*
3.7.6. Root Complex Internal Link Control *60*
3.7.7. Root Complex Event Collector Endpoint Association *60*
3.7.8. MFVC *60*
3.7.9. Virtual Channel (09h) *60*
3.7.10. RCRB *60*
3.7.11. Vendor Specific *60*
3.7.12. CAC *60*
3.7.13. ACS *60*
3.7.14. ARI *61*
3.7.15. ATS *61*

4. ERROR HANDLING 62

4.1. BASELINE ERROR REPORTING 62
4.2. ADVANCED ERROR REPORTING 63
4.2.1. VF Header Log *64*
4.3. ADVANCED ERROR REPORTING CAPABILITY 65
4.3.1. Advanced Error Reporting Enhanced Capability Header (Offset 00h) *65*
4.3.2. Uncorrectable Error Status Register (Offset 04h) *65*

4.3.3. Uncorrectable Error Mask Register (Offset 08h) ... 66
4.3.4. Uncorrectable Error Severity Register (Offset 0Ch) ... 66
4.3.5. Correctable Error Status Register (Offset 10h) ... 66
4.3.6. Correctable Error Mask Register (Offset 14h) ... 67
4.3.7. Advanced Error Capabilities and Control Register (Offset 18h) ... 67
4.3.8. Header Log Register (Offset 1Ch) ... 67
4.3.9. Root Error Command Register (Offset 2Ch) ... 68
4.3.10. Root Error Status Register (Offset 30h) ... 68
4.3.11. Correctable Error Source Identification Register (Offset 34h) ... 68
4.3.12. Error Source Identification Register (Offset 36h) ... 68

5. INTERRUPTS ... 69

5.1. REQUIREMENTS ... 69
5.1.1. PF Requirements ... 69
5.1.2. VF Requirements ... 69
5.1.3. Common PF/VF Requirements ... 70
5.2. INTERRUPT MECHANISMS ... 70
5.2.1. MSI Interrupts ... 70
5.2.2. MSI-X Interrupts ... 71
5.2.3. Address range isolation ... 71
5.2.4. MSI/MSI-X Capability field replication in VF and PF ... 71
5.2.5. MSI/MSI-X dependencies between PF and VF ... 71

6. POWER MANAGEMENT ... 72

6.1. VF DEVICE POWER MANAGEMENT STATES ... 72
6.2. LINK POWER MANAGEMENT STATE ... 72
6.3. VF POWER MANAGEMENT CAPABILITY ... 72

7. ACKNOWLEDGEMENTS ... 74

Figures

FIGURE 2-1: BAR SPACE EXAMPLE FOR SINGLE BAR DEVICE ... 20
FIGURE 2-2: INITIAL VF STATE ... 24
FIGURE 2-3: VF MIGRATION STATE DIAGRAM ... 25
FIGURE 3-1: SINGLE-ROOT IO VIRTUALIZATION EXTENDED CAPABILITIES ... 29
FIGURE 3-2: TYPE 0 CONFIGURATION SPACE HEADER ... 42
FIGURE 3-3: PCI EXPRESS CAPABILITY STRUCTURE ... 47
FIGURE 3-4: PCI EXPRESS CAPABILITY LIST REGISTER ... 47
FIGURE 3-5: PCI EXPRESS CAPABILITIES REGISTER ... 48
FIGURE 3-6: DEVICE CAPABILITIES REGISTER ... 49
FIGURE 3-7: DEVICE CONTROL REGISTER ... 50
FIGURE 3-8: DEVICE STATUS REGISTER ... 52
FIGURE 3-9: LINK CAPABILITIES REGISTER ... 53
FIGURE 3-10: LINK CONTROL REGISTER ... 54

FIGURE 3-11: LINK STATUS REGISTER 55

Tables

TABLE 3-1: SR-IOV EXTENDED CAPABILITY HEADER 30
TABLE 3-2: SR-IOV CAPABILITIES 30
TABLE 3-3: SR-IOV CONTROL 31
TABLE 3-4: SR-IOV STATUS 33
TABLE 3-5: VF MIGRATION STATE ARRAY OFFSET 38
TABLE 3-6: VF MIGRATION STATE ENTRY 38
TABLE 3-7: VF MIGRATION STATE DESCRIPTIONS 39
TABLE 3-8: SR-PCIM INITIATED VF MIGRATION STATE TRANSITIONS 39
TABLE 3-9: MR-PCIM INITIATED VF MIGRATION STATE TRANSITIONS 40
TABLE 3-10: REGISTER ATTRIBUTES (MODIFIED FROM PCIE BASE 1.0 DRAFT) 40
TABLE 3-11: COMMAND REGISTER 43
TABLE 3-12: STATUS REGISTER 44
TABLE 3-13: PCI EXPRESS CAPABILITY LIST REGISTER 47
TABLE 3-14: PCI EXPRESS CAPABILITIES REGISTER 48
TABLE 3-15: DEVICE CAPABILITIES REGISTER 49
TABLE 3-16: DEVICE CONTROL REGISTER 50
TABLE 3-17: DEVICE STATUS REGISTER 52
TABLE 3-18: LINK CAPABILITIES REGISTER 53
TABLE 3-19: LINK CONTROL REGISTER 54
TABLE 3-20: LINK STATUS REGISTER 55
TABLE 4-1: PCI EXPRESS CAPABILITIES REGISTER 65
TABLE 6-1: POWER MANAGEMENT CAPABILITIES - PMC 72
TABLE 6-2: POWER MANAGEMENT CONTROL/STATUS – PMCSR 73

Objective of the Specification

The purpose of this document is to specify PCI I/O virtualization and sharing technology. The specification is focused on single-root topologies, e.g. a single computer that supports virtualization technology.

This document is to be used in conjunction with, and does not supersede, the terms and conditions specified in the *PCI-SIG™ Trademark and Logo Usage Guidelines* document.

Document Organization

Chapter 1 provides an architectural overview of single root (SR) I/O virtualization technology and the associated methods used in this specification.

Chapter 2 specifies SR IOV Initialization and Resource Allocation

Chapter 3 specifies SR IOV configuration.

Chapter 4 specifies SR IOV error handling.

Chapter 5 specifies interrupts.

Chapter 6 specifies event handling.

Chapter 7 specifies power management.

Documentation Conventions

Capitalization

Some terms are capitalized to distinguish their definition in the context of this document from their common English meaning. Words not capitalized have their common English meaning. When terms such as "memory write" or "memory read" appear completely in lower case, they include all transactions of that type.

Register names and the names of fields and bits in registers and headers are presented with the first letter capitalized and the remainder in lower case.

Numbers and Number Bases

Hexadecimal numbers are written with a lower case "h" suffix, e.g., 0FFFFh and 80h. Hexadecimal numbers larger than four digits are represented with a space dividing each group of four digits, as in 1E FFFF FFFFh. Binary numbers are written with a lower case "b" suffix, e.g., 1001b and 10b. Binary numbers larger than four digits are written with a space dividing each group of four digits, as in 1000 0101 0010b.

All other numbers are decimal.

Reference Information

Reference information is provided in various places to assist the reader and does not represent a requirement of this document. Such references are indicated by the abbreviation "(ref)." For example, in some places, a clock that is specified to have a minimum period of 400 ps also includes the reference information maximum clock frequency of "2.5 GHz (ref)." Requirements of other specifications also appear in various places throughout this document and are marked as reference

information. Every effort has been made to guarantee that this information accurately reflects the referenced document; however, in case of a discrepancy, the original document takes precedence.

Implementation Notes

Implementation Notes should not be considered to be part of this specification. They are included for clarification and illustration only. Implementation Notes within this document are enclosed in a box and set apart from other text.

Still being defined

Text in this color denotes semantics or syntax that is still being discussed in the WG.

Terms and Abbreviations

Address Translation and Protection Table (ATPT)	The data structure(s) accessed by a Translation Agent to determine the mapping of un-translated DMA addresses into translated addresses. The data structure may contain fields that indicate the protection attributes of the translation entry. (Refer to the ATC specification for updates to the definition.)
Configuration Time VI Involvement	Refers to the operations performed by a VI on behalf of a hardware or software agent on a VE to associate it with a System Image.
Intermediary based PCI IOV	An IOV mechanism that uses a Virtualization Intermediary.
IO Virtualization	The capability for a single IO unit to be used by more than one System Image.
IOV	IO Virtualization
IOV Enabled PCIe Component	A PCIe RC, Switch, Bridge, or Endpoint that supports the required IOV extensions defined in this specification. Since MRA extensions are a superset of IOV extensions, an MRA enabled component is also an IOV enabled component.
IOV RC IOV Switch IOV Bridge IOV Endpoint	See IOV Enabled PCIe Component.
Guest Physical Address	The address used by a System Image to access system memory.
MR - Multi-Root Topology	A PCIe topology that interconnects more than one root through multi-root aware Switches.
MRA Enabled PCIe Component Multi-Root Aware Component	A PCIe RC, Switch, Bridge, or Endpoint that supports the required multi-root IOV extensions defined in this specification. MRA components are used to provide each PCIe root complex with its own virtual, PCI 32 or 64-bit Memory, IO and Configuration Space.
MRA RC, MRA Switch MRA Bridge	See MRA Enabled Component.

MRA Endpoint	
MRA VE	Virtual Endpoint (VE) in an MRA endpoint.
Native IOV	An IOV mechanism where a physical PCI Endpoint supports one or more VEs. Each VE is assigned to an SI using configuration operations that are issued by a VI. Mechanisms exist so that DMA transfers, MMIO accesses and interrupt operations can be performed directly between an SI and the VE, without Run-Time VI Involvement. The mechanism may require Run-Time VI Involvement for some operations (e.g. Configuration Space accesses).
PCI Bus Memory Address	Refers to the address portion of a PCI-* Memory Transaction
PCI Manager (PCIM)	Software or firmware that discovers the PCIe fabric. It also initializes MRA Components such that each PCIe Root complex has its own Virtual Hierarchy.
PCIe	PCI Express
PCIe RC	The PCIe Root Complex, including any Root Ports exported by that RC as defined in the PCIe documentation suite. In a virtual system, the PCIe RC may be virtualized by any combination of hardware and software.
Physical Address	The address used by the system memory controller to access system memory.
Physical Function	A PCI Function that supports the SR-IOV capabilities defined in this specification. A PF contains the SR-IOV capabilities structure.
RC	Root Complex
RP	Root Port
Run-Time VI Involvement	Refers to the intervention by a VI on operations performed between an SI and a physical PCI Endpoint (versus a VE). Note for a PCI Endpoint that supports Native IOV, mechanisms exist such that VI Run-Time Involvement may not be required.
Serial Sharing	The sequential assignment and utilization of a PCI Endpoint Device by multiple SI. Each SI has exclusive control of the Device while it is assigned.
Simultaneous Sharing	In parallel assignment and utilization of a PCI Endpoint Device by multiple SI. Each SI has exclusive control of a subset of the Device resources while sharing common resources and state, e.g. work queues are assigned per SI while the PCIe Link is shared by all SI.
SI - System Image	A software component running on a Virtual System to which specific virtual and physical Endpoints can be assigned. Specification of the behavior and architecture of an SI is outside the scope of this specification. Examples of SIs include guest operating systems and

	shared/non-shared protected domain Endpoint drivers.
SR-IOV capabilities	The SR-IOV capability structure is used to discover and configure a PF's virtualization capabilities. These virtualization capabilities include the number of Virtual Functions (VF) the PCIe Device will associate with a PF and the type of BAR mechanism supported by those VFs.
Translated Address	An address formed by using the results of an address translation request. (Refer to the ATC specification for updates to the definition.)
Translation Agent (TA)	A logical entity that converts addresses expressed in terms of one address space into an address in a different address space. A Translation Agent (TA) is associated with memory that contains translation tables that the TA will reference for address translation (See ATPT). A TA may contain an Address Translation Cache. (Refer to the ATC specification for updates to the definition.)
Un-translated Address	An address that is formed using the existing programming mechanisms of PCI Express. (Refer to the ATC specification for updates to the definition.)
Guest Virtual Address	The address of a location in virtual memory that is assigned to a user application within an SI and, typically, is protected from access by other user applications. It is typically translated to a Guest Physical Address and then to a Physical Address. The processor's address translation mechanisms generally are used to implement such mappings.
VF - Virtual Function	A Function that is associated with a Physical Function. A VF shares one or more physical resources, such as a link, with the Physical Function and other VFs that are associated with the same PF. A VF supports Native Based IOV.

1

1. Architectural Overview

1.1.1. Virtualization Technology Overview

System virtualization allows physical resources, such as processors, memory, and IO components to be shared across different SIs (SI). Therefore, virtualization allows multiple SIs to concurrently run within a platform. This results in a potential problem as the number of PCI IO devices available on a platform could be insufficient to provide each SI with its own dedicated set of PCI devices, needed to perform the functions required. Even if a sufficient number of physical PCI devices could be installed, the sharing of PCI devices is desirable, because it allows multiple SI to share an underutilized PCI device, thereby reducing overall solution costs. Additionally, as the number of devices increases, there is a corresponding increase in the external infrastructure needed for connectivity (e.g. network and storage cables, switches, routers, etc.). By reducing the number of devices required to deliver a specific performance level, PCI IO Virtualization reduces the overall solution cost.

PCI IO Virtualization can also be used to enable IO load balancing and bandwidth management across the SIs. If a SI needs more bandwidth or connectivity, then more physical device capacity can be associated with that SI. If a SI begins to consume more IO resources than it should, its consumption can be throttled back down to a preset level. These capabilities can be used by workload management personnel or automation software to apportion PCI IO virtual resources based on the urgency/importance of individual workloads.

In addition, PCI IO Virtualization can be used to increase the reliability, availability, and serviceability of a system. This is accomplished in much the same manner as with the load balancing mentioned above. That is, spare or underutilized virtual resources can be substituted for failing components.

Currently, platforms use two basic approaches to virtualize system resources - Physical Partitioning and Hypervisors. Physical Partitioning creates virtual platforms as fractions of a physical platform's resources, typically in coarse (e.g., physical) allocation units (e.g., a whole processor, along with its associated memory and IO devices). Hypervisors are software or firmware components that can virtualize all physical resources with fine granularity (e.g., in small fractions of a single physical resource). Platforms that support virtualization currently have two options for handling IO. The first option is to dedicate a physical IO device to a single virtual platform; i.e. each virtual platform gets a subset of the available PCI devices. The second option is to add a function that provides the isolation necessary to permit multiple SIs to share a single physical device into the Hypervisor, or another intermediary.

The first option has several problems. One significant problem is that devices or IO devices cannot be shared between virtual platforms. Even if a virtual platform only needs to use a fraction of the functions, bandwidth or performance of a device, the entire device must be dedicated to the specific

virtual platform that owns the device. As the number of virtual platforms on the physical platform increases, the number of devices to offer the functionality needed in each SI increases proportionately. Potentially many of those dedicated devices will be underutilized, increasing the total cost of the solution. For physical platforms that support many virtual platforms, another significant problem with this option is that it requires many device slots, with all the accompanying hardware (e.g., chips, connectors, cables, etc.) required to attach those devices to the physical platform. Adding extra PCI device slots to the design of platforms, just to accommodate the number of devices that will be required for basic functionality of virtual SIs becomes impractical and very expensive, especially when hundreds of virtual systems or partitions are supported, such as in some of today's high end physical platforms. It is very probable that the trend will be to support even more virtual systems within one physical platform in the future.

Though the second option, the use of Hypervisors or virtual intermediaries (VI), provides a mechanism for sharing devices between virtual platforms, that mechanism must be invoked and executed on every IO transaction. The invocation and execution of the sharing mechanism by the Hypervisor or other intermediary on every IO transaction degrades performance. It also leads to a more expensive solution, because the customer must purchase more hardware, either to make up for the cycles used to perform the sharing mechanism or, if the sharing mechanism is offloaded to an intermediary, for the intermediary hardware.

The IO Virtualization overhead associated with data transmission over PCI in topologies where devices are shared through intermediaries can result in a 5-7 times increase in CPU utilization vs. the case where dedicated devices are used. This hit in performance can be alleviated when natively shared devices are used because the virtual intermediaries' involvement is reduced to configuration-time only, and no virtual intermediary involvement is required during run-time

1.1.2. PCI Technologies

- This SR IOV specification builds upon *PCI Express Base Specification version 1.1*.
 - Where applicable, the specification notes relevant deltas between the *PCI Express Base Specification version 1.1* and *2.0*.
 - In general, the SR IOV specification does not touch the physical, data link, or the transaction layers specified within the *PCI Express Base Specification*. The specification primarily impacts configuration space and how resources are managed and events handled.
- I/O virtualization capabilities communicated through new PCI Express capabilities
 - Capabilities exist in PCI Express extended configuration space
- SR IOV specification has no impact on the PCI or PCI-X specifications.
 - PCI and PCI-X components can be intermixed with SR IOV components. The IOV robustness and capabilities will vary as a result.

1.1.3. Interoperability Requirements

- All SR IOV components are required to interoperate with components designed in compliance with the *PCI Express Base Specification version 1.1*.

- SR IOV components are required to interoperate with non-SR IOV capable software. SR IOV solution robustness may be reduced as a result.

- SR IOV components may optionally be designed in compliance with the *PCI Express Specification version 2.0*.
- A hierarchy can be composed of a mix of SR IOV and non-SR IOV components. For example, a hierarchy may contain any mix of SR IOV and non-SR IOV Endpoint Devices.
- A hierarchy can be composed of a mix of PCIe and PCI Express to PCI / PCI-X Bridges.
 - A PCI Express to PCI / PCI-X Bridge and PCI / PCI-X Devices can be serially shared by multiple System Images (SI).

1.1.4. Related Technologies Outside of Scope

- Discussion ATPT technology
 - Address Translation and Protection Table
 - Translates and protects system memory accessed through the PCI bus.
 - Consists of the following functions:
 - On incoming DMA from a PCI Function, translates the PCI Bus Address into an address that can be used to directly access host memory.
 - Provides physical Function isolation by assuring that when a PCIe function performs a DMA, the operation targets a System Image (SI) associated with that function.
 - ATPT technology is outside the scope of the PCI-SIG
- Discussion of Interrupt Mapping
 - Routes PCI interrupts to the SI associated with the PCI Requester ID that generated the interrupt.
 - Interrupt mapping technology is outside the scope of the PCI-SIG
- Brief discussion of Address Translation Services (ATS) technology
 - The PCI-SIG ATS specification enables a PCIe Function to reduce the resource pressures and potential performance impacts associated with an ATPT.
 - ATS may be optionally implemented on a per Function basis. Designs can support ATS on zero or more Functions within an Endpoint Device.
 - ATS is not required to support SR IOV but a design that supports ATS may yield a higher performing solution.

1.2. Example Usage Models

- Description of client – desktop, mobile, etc.

- Description of server – point-to-point, PCIe Switch, PCIe to PCI / PCI-X Bridge topologies
- Description of storage – point-to-point, PCIe Switch, PCIe to PCI / PCI-X Bridge topologies
- Description of embedded

Each description will provide examples of how IOV technology can be applied, the requirements that stem from a usage model, serial vs. simultaneous sharing, and the value that can be derived by use of IOV technology. These descriptions will be added in a subsequent version of the draft specification.

1.3. Single Root (SR) Architectural Elements

<Will incorporate elements from various sections. This may entail movement of informative text from the subsequent sections into a single area to consolidate information and simplify the specification. This will be done in a subsequent draft of the specification.>

Minimum set of elements that will be covered:

Physical Function (PF)

- Equivalent to the PCI Express Base Specification Function.
- An Endpoint Device contains one or more PF
- Each PF has a unique BAR and a full configuration space
 - All configuration accesses are performed by a VI; SI direct access is not allowed
- Each PF may support zero or more VF
- Each PF can be serially shared by multiple SI

Virtual Function (VF)

- Does not have a full configuration space
 - All configuration accesses are performed by a VI; SI direct access is not allowed
- Each VF can be serially shared by multiple SI

Single Root Hierarchy

- Single Root refers to a Root Port (RP) within a Root Complex (RC). Each RP represents a separate hierarchy.

Single Root PCI Configuration Manager (PCIM)

- Responsible for managing the SR IOV configuration space and resources.
- Responsible for resource assignment and arbitration
- Responsible for event handling and prorogation where applicable to individual SI
 - Errors, Power Management, Hot-Plug, etc.

1.4. Example Operational Flows

<Will add a set of example flows to illustrate how configuration, events, and main data path operations occur.>

1.5. TBD

2

2. Initialization and Resource Allocation

Initially and after a conventional reset, a PCIe Function that supports the SR-IOV capabilities defined in this specification shall have the SR-IOV capabilities disabled. In this specification each Function that supports the SR-IOV capabilities is referred to as a Physical Function (PF). A Function that doesn't support the SR-IOV capabilities is referred to as a Function. A PF contains the SR-IOV capabilities structure. The SR-IOV capability structure is used to discover and configure the PF's virtualization capabilities. These virtualization capabilities include the number of Virtual Functions (VF) the PCIe Device will associate with a PF and the type of BAR mechanism supported by those VFs. A more complete list of these capabilities will be described later in this chapter. The fields used to configure these capabilities will be described in the next chapter.

2.1. SR-IOV Resource Discovery

The following sections describe how SW determines that a device is SR-IOV capable and subsequently identifies VF resources through Virtual Function Configuration Space.

2.1.1. SR-IOV Capability Discovery

The PF's SR-IOV Extended Capability is used to discover and configure the virtualization capabilities of the VFs associated with that PF. The SR-IOV Extended Capability must be present in each Physical Function compliant with this specification. After initialization or conventional reset each Physical Function of an SR-IOV capable PCIe Device has its SR-IOV capabilities disabled. The PF's SR-IOV Extended Capability is described in section 3.3 SR-IOV Extended Capability.

A Device that supports multiple PFs must have an SR-IOV Extended Capability for each PF. A Device may have a mix of Functions and PFs, where the Functions do not support the SR-IOV capabilities defined in this specification. Each Function and PF consumes one Requestor Identifier (RID).

Functions and PFs must be located using Base PCIe rules. The Device may or may not support the ARI Capability. If the Device supports the ARI Capability, the Device may support more than 8 Functions and PFs as defined by the ARI Capability. The InitialVFs and TotalVFs fields are used to discover the maximum number of VFs that can be associated with a PF. If the Device does not support VF migration, TotalVFs and InitialVFs shall contain the same value.

If the Device supports VF migration, when TotalVFs is read, the PF must return the number of VFs that can be assigned to the PF. For such a Device, when InitialVFs is read, the PF must return the initial number of VFs assigned to the PF. The TotalVFs and InitialVFs fields are described in sections 3.3.6and 3.3.5.

The PF has its own independent set of BARs in its standard configuration space and a Memory Space Enable bit to control that space. The VFs share a BAR set and have a Memory Space Enable bit that controls the memory space of all the VFs. That is, if the MSE bit is disabled, the memory mapped space allocated for all VFs is disabled. Additionally, the BAR set that is shared by all the VFs resides in the PF's SR-IOV capabilities. For a description of how the VF BAR is configured see section 2.1.2.1. For a description of the VF BAR SR-IOV capabilities see section 3.3.13.

Prior to configuring the SR-IOV Capabilities, the Supported Page Sizes configuration field is used to discover the page sizes supported by the PF and its associated VFs. When this field is read, the PF must return the page sizes it can support. Section 2.1.2.1 describes how this field can be used to align a PF or VF's MMIO aperture on a system page boundary. This Capabilities field is described in section 3.3.11.

2.1.2. Configuring SR-IOV Capabilities

This section describes the fields that must be configured before enabling a PF's IOV capabilities. The PF's IOV capabilities are enabled by writing a "1" to the "VF Enable" bit. As with the base PCIe Specification, the Device's VFs may return CRS responses to configuration requests after a "1" is written to the VF Enable bit to indicate it is not done reconfiguring its internal resources. The VF Enable bit is described in section 3.3.3.1.

The Num VFs field defines the number of VFs the PCIe Device must associate with the PF when the PF's IOV capabilities are enabled. The NumVFs field is described in section 3.3.7.

2.1.2.1. Configuring the BAR Mechanisms

The System Page Size field defines the page size the system will use to map the PF and VF's PCIe memory addresses when the PF's IOV capabilities are enabled. The value chosen for the System Page Size must be one of the Supported Page Sizes. The System Page Size field is used by the Device to align the MMIO aperture defined by each BAR to a system page boundary. The System Page Size field is described in section 3.3.12.

The behavior of VF BARs is the same as the PCI Local Bus Specification 3.0 normal PCI BARs, except that a VF BAR describes the aperture for each VF, whereas a PCI BAR describes the aperture for a single function:

- The behavior described in the PCI Local Bus Specification 3.0 for determining the memory aperture of a Function's BAR applies to each VF BAR. That is, the size of the memory aperture required for each VF BAR can be determined by writing all "1"s and then reading the VF BAR. The results read back must be interpreted as described in the PCI Local Bus Specification.
- The behavior for assigning the starting memory space address of each BAR associated with the first VF is also as described in the PCI Local Bus Specification 3.0. That is, the address written into each VF BAR is used by the Device to set the starting address for that BAR on the first VF.
- The difference between VF BARs and PCI Local Bus Specification 3.0 BARs is that for each VF BAR, the memory space associated with the second and higher VFs is derived

from the starting address of the first VF and the memory space aperture. That is, BAR1 VF N SA = BAR1 VF 1 SA + (N – 1) x (BAR1 VF MA); where BAR1 VF 1 SA is the address written into VF BAR1 and BAR1 VF MA is the memory aperture of VF BAR1. Additionally, VF memory space is not enabled until both VF Enable and VF MSE have been Set. Note reading the BARs before software has configured the PF's System Page Size may not return the final aperture, because the PF may change its aperture based on the System Page Size value.

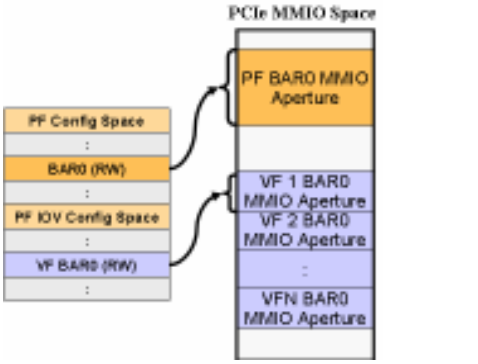


Figure 2-1: BAR Space Example for Single BAR Device

2.1.3. VF Discovery

After the PF's IOV capabilities are enabled, the VFs associated with a PF are discoverable by reading the First VF Offset and VF Stride fields in the PF's SR-IOV Extended Capabilities. The First VF Offset and VF Stride fields are 16-bit Requestor Identifier (RID) offsets. These offsets are used to compute the RIDs for the VFs as shown in the table below.

VF Number	VF RID
VF 1	(PF RID + VF Offset) Modulo 2^{16}
VF 2	(PF RID + VF Offset + VF Stride) Modulo 2^{16}
VF 3	(PF RID + VF Offset +2 * VF Stride) Modulo 2^{16}
…	…
VF N	(PF RID + VF Offset +(N-1) * VF Stride) Modulo 2^{16}
…	…
VF NumVFs (last one)	(PF RID + VF Offset +(NumVFs-1) * VF Stride) Modulo 2^{16}

All arithmetic used in this RID computation is 16 bit unsigned dropping all carrys. Note for some VF Stride values, this can result in VF 2 having a RID value that is numerically larger than VF 3.

All VFs and PFs must have distinct Requestor IDs. The RID of any PF or VF must not overlap with the RID of any other PF or VF given any valid setting of NumVFs across all PFs of a device.

VF Stride and First VF Offset are constants. Their values may be affected by the VF ARI Enable bit but their values are not affected by other settings in this PF or any other PF of the Device.

VFs may reside on different bus number(s) than the associated PF. This can occur if, for example, First VF Offset has the value 0100h. No VF may be located on a bus number that is numerically smaller than its associated PF.

As in Base PCIe, SR-IOV capable Devices capture the bus number from any Type 0 config request. SR-IOV capable Devices do not capture the bus number from any Type 1 config requests.

Switch processing of bus numbers is unchanged from base PCIe. In Base PCIe, the switch sends all config requests in the range Secondary Bus Number (inclusive) to Subordinate Bus Number (inclusive) to the Device. Type 1 config requests addressing the Secondary Bus Number are converted into Type 0 config requests while Type 1 config requests addressing bus numbers between Secondary Bus Number (exclusive) and Subordinate Bus Number (inclusive) are forwarded on to the Device as Type 1 config requests.

If the VF ARI Enable is Set in the Device, then RIDs sent to the Device on the captured Bus Number are interpreted as an 8 bit Bus Number and an 8 bit Function Number (and there is no Device Number). Additional Bus Numbers may be used to assign VFs RIDs sent to the Device on these additional Bus Numbers are always interpreted as an 8 bit Bus Number and an 8 bit Function Number independent of the value of VF ARI Enable. VF ARI Enable is read only with VF Enable is Set.

If the VF ARI Enable is Clear in the Device, then there are only 8 Functions on the first Bus Number (i.e. the Device number field must be zero).

All PFs must be located on the Device's captured Bus Number. As noted above, VFs may be located on additional bus numbers. Bus Numbers are a constrained resource. Devices are strongly encouraged to avoid leaving "holes" in their Bus Number usage so as to avoid wasting Bus Numbers.

IMPLEMENTATION NOTE

As an example, consider an SR-IOV Device that supports a single PF. Initially, only PF 0 is visible. Software Sets VF ARI Enable. From the SR-IOV Capability it determines: InitialVFs is 600, First VF Offset is 1 and VF Stride is 1.

- If software sets NumVFs in the range [0 .. 255], then the Device uses a single Bus Number.
- If software sets NumVFs in the range [256 .. 511], then the Device uses two Bus Numbers.
- If software sets NumVFs in the range [512 .. 600], then the Device uses three Bus Numbers.

PF 0 and VF 0,1 thru VF 0,255 are always on the first (captured) bus number. VF 0,256 thru VF 0,511 are always on the second Bus Number (captured Bus plus 1). VF 0,512 thru VF 0,600 are always on the third Bus Number (captured Bus plus 2).

Software should configure switch Secondary and Subordinate Bus Number fields to route enough bus numbers to the Device. If sufficient Bus Numbers are not available, software should reduce a

Device's bus number requirements by not enabling SR-IOV and/or reducing NumVFs for some or all PFs of the Device prior to enabling SR-IOV.

As stated previously, when the PF's SR-IOV capabilities have been enabled the Device must assign NumVFs to the associated PF. NumVFs is Read Only when the PF's VF Enable is Set.

After the PF's SR-IOV capabilities are enabled, each Active VF must respond to PCIe transactions that target VF Configuration space fields defined in sections 3.4 to 3.7. Unless VF Migration is enabled, all VFs are Active. For details on VF Migration, see sections 2.4.

2.1.4. Function Dependency Lists

PCI Devices may have Vendor Specific dependencies between Functions. For example, Functions 0 and 1 might provide different mechanisms for controlling the same underlying hardware. In such situations, the Device programming model might require that these dependent Functions be assigned to SIs as a set.

Function Dependency Lists are used to inform SR-PCIM about a Device's Function dependencies (or to indicate that there are no Function dependencies). SR-PCIM must then assign PFs and VFs to SIs such that the dependencies are satisfied.

See Section 3.3.8 for details.

2.1.5. Interrupt Resource Allocation

PFs and VFs are required to support either MSI, MSI-X interrupts or both. VFs do not support INTx delivery of interrupts.

2.1.6. Resources Not Supported List

When any PF of a PCIe Device has its IOV capabilities enabled:

- Phantom functions must not be enabled on the Device. If phantom functions are enabled, the behavior is Undefined.
- The PF or VF must not use I/O mapped BARs.
- INTx must not be used.

2.2. Reset Mechanisms

This section describes how PCI base defined reset mechanisms affect Devices that support SR-IOV. It also describes the mechanisms used to reset a single VF and a single PF with its associated VFs.

2.2.1. Conventional Reset

A Fundamental or Hot Reset to a Device that supports this specification shall cause all Functions, PFs, and VFs context to be reset to their original, power-on state.

If a Device's PF has its SR-IOV capabilities enabled and a Fundamental or Hot Reset is issued to the Device, the Device must reset all PF and VF state:

- The PF must have its SR-IOV capabilities disabled and reverts back to being a PCI Function. All SR-IOV capabilities that are settable (e.g. NumVFs) are reset to their default values.
- MSE and BME are both off.
- All BAR values used by the PF and VFs are indeterminate.
- All interrupts are disabled.
-

2.2.2. FLR that targets a VF

An FLR that targets a VF must be supported as defined by the PCI Base Specification.

Note: Software may use FLR to reset a VF as described by the PCI Base Specification. FLR does not affect the VFs existence or any address assigned to it. A VFs BAR registers and VF MSE (See section 3.x.x.x) are unaffected by FLR.

2.2.3. FLR that targets a PF

An FLR that targets a PF must be supported as defined by the PCI Base Specification including resetting the SR-IOV Capability.

Note: VF Enable will be Cleared after an FLR to the PF which means that VFs are no longer allocated or enabled.

2.3. IOV Re-initialization and Reallocation

If the PF's IOV capabilities are disabled after having been enabled, all of the PF's VFs are effectively destroyed and must no longer: issue PCIe transactions, exist in the RID space or exist in the memory mapped address space. The behavior must be as if an FLR was issued to each of the VFs. Specifically, VFs must not retain any context after the PF's IOV capabilities have been disabled. Any errors already logged via PF error reporting registers, remain logged. However, no new VF errors shall be logged after the PF's IOV capabilities are disabled, except as permitted by FLR rules.

2.4. VF Migration

In Multi-Root systems, VFs can be migrated between Virtual Hierarchies. VF Migration does not occur in Single-Root only systems.

SR support for VF Migration is *Optional*. VF Migration must only occur if it has been enabled by through the PF's SR-IOV capabilities by SR software. If migration operations are disabled in the PF's SR-IOV capabilities, VFs are always Active, and the VF State array is undefined.

Supporting VF Migration places the following requirements on SR software:

- The need to determine if a VF is *Active* or *Inactive*. A VF that is active is available for use by SR. That is, an Active VF must have its PCI configuration and memory mapped address space accessible. An Inactive VFs must not be available for use by SR. That is, an Inactive VF consumes Memory and Configuration address space but cannot be accessed in either space.
- The need to participate in *Migrate In* operations. A Migrate In operation is used to offer a VF to SR. A Migrate In operation is initiated by MR-PCIM. SR software can accept a Migrate In operation and move a VF to the *Active.Available* state.
- The need to participate in *Migrate Out* operations. A Migrate Out operation is used to request the graceful removal of an Active VF from use by SR. SR Software can accept the Migrate Out and move the VF to the *Inactive.Unavailable* state.
- The need to handle *Forced Completions of Migrate Out* operations. A Force Completion of a Migrate Out is when MR-PCIM accepts a Migrate Out operation instead of SR software. This is the VF Migration equivalent of a Surprise Down on in Base PCIe.
- The need to handle *Migrate In Retractions*. This is when an offer of a VF to SR is retracted by MR-PCIM and the VF moves back to the *Inactive.Unavailable* state.

Initial VF State is shown Figure 2-2 below. VF 1 thru InitialVFs (inclusive) initialize in the Active.Available state and VF InitialVFs (exclusive) thru TotalVFs initialize in the Inactive.Unavailable state. If VF Migration is not enabled, VFs above InitialVFs are not used and no migrations occur so that VFs below InitialVFs remain in Active.Available.

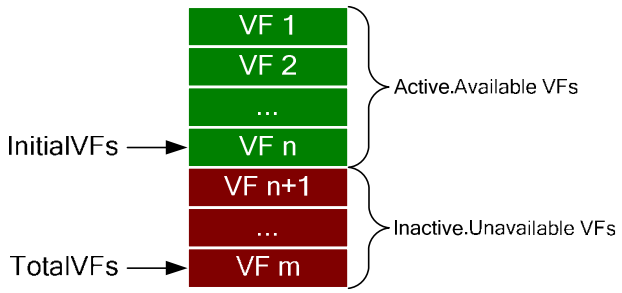


Figure 2-2: Initial VF State

VF migration follows the Figure 2-3 below. The state values shown are contained in the VF State array entry associated with the VF. State transitions indicated by solid lines are initiated by MR software. State transitions indicated by dashed lines are initiated by SR software.

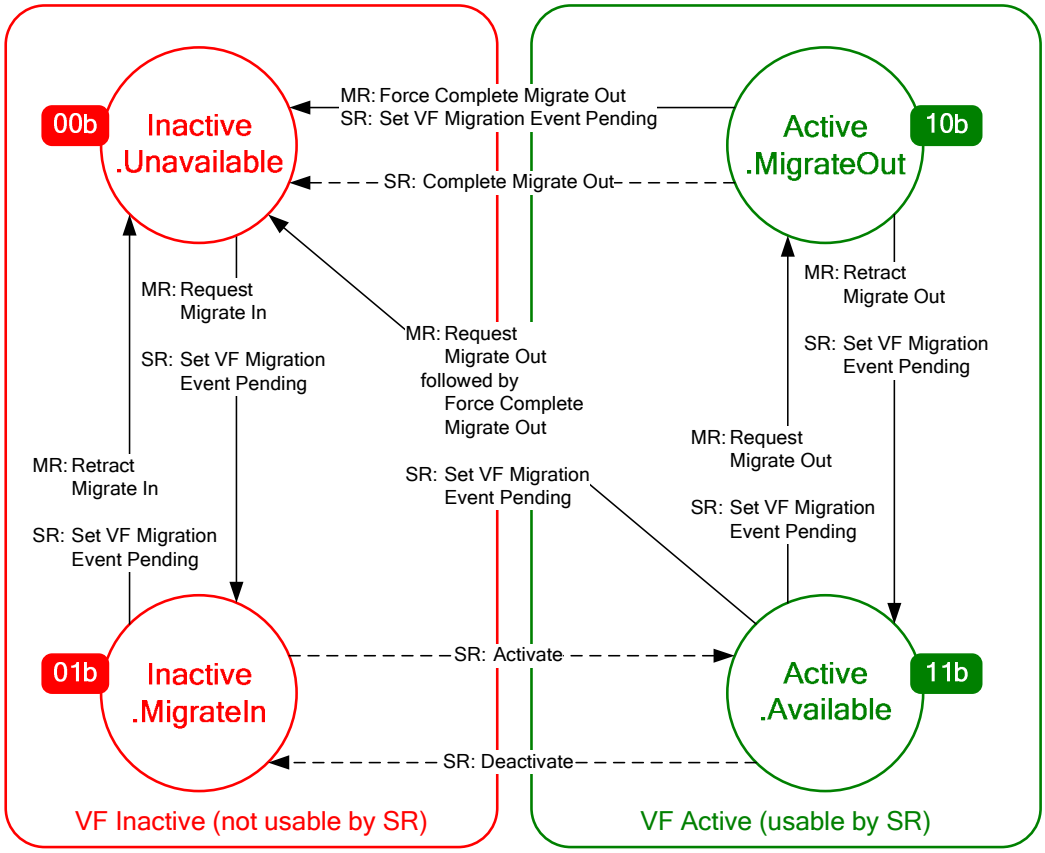


Figure 2-3: VF Migration State Diagram

SR software initiates a state transition by writing a new value to the VF Migration State array. Only the following state transitions may be initiated by SR software. Writes by SR software attempting to initiate any other state transition are ignored and no state transition occurs.

Current State	Written State	Meaning
01b	11b	SR Activate VF
11b	01b	SR Deactivate VF
10b	00b	SR Complete Migrate Out

Implementation Note: Typical SR-PCIM usage will be to (1) read the VF Migration State, (2) if the current VF Migration State is as expected, write the new VF Migration State value, and (3) re-read

the VF Migration State to verify that the state change took place. If the VF Migration State changed after step 1, the write in step 2 might be ignored.

VFs in states Inactive.Unavailable or Inactive.MigrateIn are not usable by software in any way. Requests targeting an Inactive VF are treated as a Completer Abort. Within 100ms milliseconds of transitioning to an Inactive state, the Device must ensure that no new transactions will be issued using the indicated RID.

State transitions initiated by MR-PCIM cause the VF Migration Event Pending bit to be set.

This migration state machine exists for every VF that supports VF Migration. Migration state machines are not affected by Function Dependency Lists (see Sections 2.1.4 and 3.3.8).

3

3.Configuration

3.1. Requirements

The following sections list the configuration requirements for Physical Functions (PFs) and Virtual Functions (VFs).

Physical Functions must be discoverable in configuration space, as with non-virtualized Functions. Physical Functions contain the SR IOV Extended Capability described in section 3.3. Physical Functions are used to discover, configure (create) and manage the Virtual Functions associated with the Physical Function and for other things described in this specification.

PFs and VFs use the standard PCI Express defined configuration mechanisms and header types, however, not all configuration fields are required to be implemented in VFs. Some fields may be reproduced from the PF as read-only copies. Some configuration space fields in VFs may require VI emulation and some configuration space fields are required to be implemented per VF. The sections 3.2 through 3.7 list these specific requirements.

Implementation note: In a virtualized system, it may be necessary to prevent direct access to configuration space from an SI. To achieve this, a VI may be required. The VI would intercede when an SI attempted to access configuration space, insuring that the SI does not change the system configuration in a way that would compromise the operation or security of other SIs.

3.1.1. PF Requirements

A PF compliant with this specification shall implement the SR-IOV Extended Capability defined in section 3.3 and the PF Configuration space fields defined sections 3.4 to 3.7.

3.1.2. VF Requirements

A VF compliant with this specification shall implement configuration space fields as defined in this chapter.

3.1.3. Virtual Intermediary (VI) Requirements

For system implementations that support more than one SI, a VI is expected to protect the fabric and to emulate certain fields in a VFs configuration space as described in sections 3.2 through 3.8.

System implementations with VIs that support multiple SIs may permit configuration space access to functions “assigned” to an SI, and block configuration space access to functions that are “not

assigned" to an SI. These "system specific" VF configuration space protection methods are beyond the scope of this specification.

The VI is also expected to emulate certain configuration space fields in VFs. In order to guarantee identical PCI Express semantics as the fabric when an SI issues a configuration read/write, the VI must emulate the PCI Express ordering rules.

There are several possible implementations of the interfaces between the SI and the VI. These implementations are outside the scope of this specification. The following describes a couple of possibilities:

- SI issues configuration operations through the VI.

 Under this implementation example, if an SI issues a configuration read or configuration write to any field in configuration space, and the VI permits that access, the VI must issue a configuration read or configuration write transaction targeting that function in the fabric, even if the field is emulated by the VI. The configuration read/write transaction must be issued prior to returning to the SI. If the transaction is a configuration read, the VI must not return to the SI until the configuration read transaction targeting the function completes at the root complex. If the transaction is a configuration write, the VI must issue a configuration write transaction targeting the function prior to returning to the SI.

- SI invokes higher level operations through the VI.

 Under this implementation example, the SI to VI interface is defined at a higher level than PCI Express configuration operations. When the SI invokes one of these higher level functions, the VI performs the necessary PCI Express configuration read or configuration write operation(s) required to complete the higher level operation.

Comment [drw1]: I don't believe that any of this should survive, even the "guarantee identical PCI Express semantics". We may design VFs with a notion of how a VI might use the facilities but how a VF is used by an SI is not something this spec can mandate.

Note: We have to decide if there's anything normative in this section

3.1.4. SR-PCIM Requirements

SR-PCIM is used in this specification to refer to the system software that controls configuration, management and error handling of PFs and VFs. The exact implementation of SR-PCIM is not defined by this specification.

Implementation Note: This specification does not state whether SR-PCIM must be implemented as a single entity or as multiple entities. This specification also does not define where SR-PCIM resides. For example, this specification does not state whether SR-PCIM is a stand-alone component or a sub-component that is embedded into a higher level component, such as the VI. This specification only defines the semantic requirements that SR-PCIM must support and the syntax and semantics of PCI Express extended configuration space fields which SR-PCIM must use to configure and manage IOV EPs.

3.2. Configuration Space

PFs compliant with this specification shall implement the SR-IOV Capability as defined in the following sections. VFs compliant with this specification shall implement configuration space fields and capabilities as defined in the following sections.

3.3. SR-IOV Extended Capability

This specification defines a new PCIe Extended Capability for each PF that supports Single Root IOV. This Capability is used to describe and control a PF's SR-IOV capabilities.

For multi-function devices, each PF that supports SR_IOV shall provide the Capability structure defined in this section.

The following figure shows the SR-IOV Capability structure.

31 24	23 20	19 16	15 0	Byte Offset
Next Capability Pointer		Capability Version	Capability ID	00h
SR IOV Capabilities				04h
SR IOV Status			SR IOV Control	08h
TotalVFs (RO)			InitialVFs (RO)	0Ch
ReservedZ	Function Dependency Link (RO)		NumVFs (RW)	10h
VF Stride (RO)			First VF Offset (RO)	14h
Supported Page Sizes (RO)				18h
System Page Size (RW)				1Ch
VF BAR0 (RW)				20h
VF BAR1 (RW)				24h
VF BAR2 (RW)				28h
VF BAR3 (RW)				2Ch
VF BAR4 (RW)				30h
VF BAR5 (RW)				34h
VF Migration State Array Offset (RO)				38h

Figure 3-1: Single-Root IO Virtualization Extended Capabilities

3.3.1. SR-IOV Extended Capability Header (00h)

Table 3-1 defines the SR-IOV Extended Capability header. The Capability ID for the SR-IOV Extended Capability is xxxxh [TBD].

Table 3-1: SR-IOV Extended Capability Header

Bit Location	Register Description	Attributes
15:0	**PCI Express Extended Capability ID** – This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. The Extended Capability ID for the SR-IOV Extended Capability is xxxh [TBD].	RO
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification.	RO
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of Capabilities. For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than 0FFh.	RO

3.3.2. SR-IOV Capabilities (04h)

Table 3-2 defines the layout of the SR-IOV Capabilities field.

Table 3-2: SR-IOV Capabilities

Bit Location	Register Description	Attributes
0	**VF Migration Capable** – If Set, the PF is Migration Capable and operating under a Migration Capable MR-PCIM.	RO
20 .. 1	**Reserved** – These fields are currently reserved	RsvdZ
31 .. 21	**VF Migration Interrupt Message Number** – Indicates the MSI/MSI-X vector used for migration interrupts. The value in this field has no significance if bit 0 of this capability is Clear.	RO

3.3.2.1. VF Migration Capable

VF Migration Capable must be Set if the PF supports VF Migration. If Clear, the PF does not support VF Migration.

VF Migration support is an optional feature of MR-IOV. If the PF does not implement this feature, this bit shall be implemented as RO value zero.

This bit is set when VF Migration support is implemented and MR-PCIM has enabled its use.

Devices that implement Single Root IOV only shall implement this field as RO value zero.

3.3.2.2. VF Migration Interrupt Message Number

VF Migration Interrupt Message Number must contain the MSI or MSI-X vector number used for the interrupt message generated in association with certain VF migration events.

For MSI, VF Migration Interrupt Message Number must indicate the MSI message number [0 .. 31] used to reference certain SR events described in this chapter. The Function is required to update this field so that it is correct if the number of MSI Messages assigned to the Function changes when software writes to the Multiple Message Enable field in the MSI Message Control register.

For MSI-X, VF Migration Interrupt Message Number must indicate the MSI-X Table entry [0 .. 2047] used to generate the interrupt message.

If both MSI and MSI-X are implemented, they are permitted to use different vectors, though software is permitted to enable only one mechanism at a time. If both MSI and MSI-X interrupts are enabled, this field is undefined.

If VF Migration Capable is Clear, this field is undefined.

3.3.3. SR-IOV Control (08h)

Table 3-3 defines the layout of the SR-IOV Control field.

Table 3-3: SR-IOV Control

Bit Location	Register Description	Attributes
0	**VF Enable** – Enables / Disables VFs. Default value is 0b	RW
1	**VF Migration Enable** – Enables / Disables VF Migration Support. Default value is 0b	RW
2	**VF Migration Interrupt Enable** – Enables / Disables VF Migration State Change Interrupt. Default value is 0b.	RW
3	**VF MSE** – Memory Space Enable for Virtual Functions. Default value is 0b.	RW
4	**VF ARI Enable** – Device may locate VFs in Function numbers 8 to 255 of the captured Bus number. Default value is 0b.	RW
15..5	**Reserved** – These fields are currently reserved	RsvdP

3.3.3.1. VF Enable

VF Enable manages the assignment of VFs to the associated PF. If VF Enable is Set, the VFs associated with the PF are enabled and are accessible in the PCI Express fabric. When enabled, VFs respond to and may issue PCI-Express transactions following the rules for PCI-Express Endpoints Functions.

If Clear, VFs are disabled and not visible in the PCI-Express fabric; VFs may not respond to or issue PCI Express transactions.

Sections 3.3.7 NumVFs, 3.3.5 InitialVFs, 3.3.6 TotalVFs, 3.3.9 First VF Offset, 3.3.12 System Page Size and 3.3.13 VF BARx and describe additional semantics associated with this field.

3.3.3.2. VF Migration Enable

VF Migration Enable must be Set to allow VF Migration on this PF.

If VF Migration Capable is Set, this bit is RW, default 0b. Otherwise, this bit is RO Zero.

This field is RO when VF Enable is Set.

3.3.3.3. VF Migration Interrupt Enable

If VF Migration Interrupt Enable is Set, a 0 to 1 transition of VF Migration Event Pending causes the PF to send an MSI interrupt message as defined by the SR IOV Interrupt Message Number field, described above. If the interrupt message cannot be sent, and MSI or MSI-X is enabled, the corresponding MSI or MSI-X pending bit is Set, if it exists.

Note: VF Migration events are described in section 2.4.

3.3.3.4. VF MSE (Memory Space Enable)

VF MSE controls memory space enable for all VFs associated with this PF, as with the Memory Space Enable bit in a functions PCI command register. The default value for this bit is 0b.

When VF Enable is Set, VF memory space access is permitted only when VF MSE is Set. VFs shall follow the same error reporting rules as defined in the base specification if an attempt is made to access a virtual functions memory space when VF Enable is Set and VF MSE is Clear.

Implementation Note: VF memory space cannot be accessed when VF Enable is Clear. Thus, VF MSE is "don't care" when VF Enable is Clear; however, software may choose to Set VF MSE after programming the VF BAR*n* registers, prior to Setting VF Enable.

3.3.3.5. VF ARI Enable

VF ARI Enable controls the assignment of Function numbers to VFs on the captured Bus number.

When VF ARI Enable is Set, VFs on the captured Bus Number may be located at any Function number (0 to 255). When VF ARI Enable is Clear, VFs on the captured Bus Number may only be located in Functions 0 to 7.

Functions on additional Bus Numbers may always be located at any Function number (0 to 255).

A Device may use the setting of VF ARI Enable to determine the values for First VF Offset and VF Stride. Doing this may help a Device conserve its usage of Bus Numbers.

Software should set this bit to match the ARI Forwarding Enable bit in the Downstream port of the switch immediately above the Device.

VF ARI Enable is only present in the lowest numbered PF of a Device. This VF ARI Enable field affects all PFs of the Device. VF ARI Enable is Read Only Zero in other PFs of a Device.

The effect of changing VF ARI Enable is Undefined when VF Enable is Set in any PF. This bit is not affected by FLR of any PF or VF.

3.3.4. SR-IOV Status (0Ah)

Table 3-4 defines the layout of the SR-IOV Status field.

Table 3-4: SR-IOV Status

Bit Location	Register Description	Attributes
0	**VF Migration Event Pending** – Indicates a VF Migration In or Migration Out Request has been issued by MR-PCIM. To determine the cause of the event, software may scan the VF State Array.	RW1C
15..1	**Reserved** – These fields are currently reserved	ReservedZ

3.3.4.1. VF Migration Event Pending

VF Migration Event Pending is Set when VF Enable, VF Migration Capable, and VF Migration Enable are Set and MR-PCIM initiates specific VF Migration State transitions (see sections 2.4 and 3.3.14.1 for details.)

This setting of VF Migration Event Pending is not affected by the value of VF Migration Interrupt Enable or by any controls for MSI or MSI-X.

If VF Migration Interrupt Enable is Set when this bit transitions from 0b to 1b, the PF will send the MSI or MSI-X interrupt defined by the SR-IOV Interrupt Message field, described above.

3.3.5. InitialVFs (0Ch)

InitialVFs indicates the number of VFs that are initially associated with the PF. If VF Migration Capable is Clear, this field must contain the same value as TotalVFs.

3.3.6. TotalVFs (0Eh)

TotalVFs indicates the maximum number of VFs that could be associated with the PF.

If VF Migration Capable is Clear, this field must contain the same value as InitialVFs. If VF Migration Capable is Set, this field must contain a value that is greater than or equal to the value of InitialVFs.

3.3.7. NumVFs (10h)

NumVFs defines the number of VFs software has assigned to the PF. Software sets NumVFs as part of the process of creating VFs. *NumVFs* VFs shall be *visible in the PCI-Express fabric* after both NumVFs is set to a valid value and VF Enable is set to one. "Visible in the PCI-Express fabric" means that the VF shall respond to PCI Express transactions targeting the VF, following all other rules defined by this specification and the base specification.

The results are undefined if NumVFs is set to a value greater than TotalVFs.

NumVFs may only be written while VF Enable is Clear. The NumVFs field is Read Only when VF Enable is Set.

3.3.8. Function Dependency Link (12h)

The programming model for a Device may have Vendor Specific dependencies between sets of Functions. The Function Dependency Link field is used to inform SR-PCIM about these dependencies.

This field describes dependencies between PFs. VF dependencies are the same as the dependencies of their associated PFs.

If a PF is independent from other PFs of a Device, this field shall contain the Function Number of the PF.

If a PF is dependent on other PFs of a Device, this field shall contain the Function Number of the next PF in the same Function Dependency List. The last PF in a Function Dependency List shall contain the Function Number of the first PF in the Function Dependency List.

SR-PCIM must allocate VFs to SIs such that if a particular SI is assigned VF *m* associated with PF *n*, that SI is also assigned VF *m* of all PFs that are in the same Function Dependency List as PF *n*.

For example, consider the following scenario:

SR-IOV Field	PF 0	PF 1	PF 2
Function Dependency Link	1	0	2
NumVFs	4	4	6
FirstVF	4	4	4
VFStride	3	3	3

Function	Description	Independent
0	PF 0	No
1	PF 1	No
2	PF 2	Yes
3	Function not present	
4	VF 0,1 (aka PF 0 VF 1)	No
5	VF 1,1 (aka PF 1 VF 1)	No
6	VF 2,1 (aka PF 2 VF 1)	Yes
7	VF 0,2	No
8	VF 1,2	No
9	VF 2,2	Yes
10	VF 0,3	No
11	VF 1,3	No
12	VF 2,3	Yes
13	VF 0,4	No
14	VF 1,4	No
15	VF 2,4	Yes
16 to 17	Functions not present	
18	VF 2,5	Yes
19 to 20	Functions not present	
21	VF 2,6	Yes
22 to 255	Functions not present	

In this example, Functions 4 and 5 must be assigned to the same SI. Similarly, Functions 7 and 8, 10 and 11, 13 and 14 must be assigned together. If PFs are assigned to SIs, Functions 0 and 1 must be assigned together as well. Functions 2, 6, 9, 12, 15, 18 and 21 are independent and may be assigned to SIs in any fashion.

All PFs in a Function Dependency List shall have the same values for the InitialVFs, TotalVFs and VF Migration Capable fields.

SR-PCIM shall ensure that all PFs in a Function Dependency List have the same values for NumVFs, VF Enabled, VF Migration Enabled fields before any VF in that Function Dependency List is assigned to an SI.

VF Migration and VF Mapping operations occur independently for every VF. SR-PCIM shall not assign a VF to an SI until it can assign all dependent VFs. For example, using the scenario above, if a both VF 0,2 (Function 7) and VF 1,2 (Function 8) are in the Inactive.Unavailable or Inactive.MigrateIn states, SR-PCIM may not assign either VF to an SI until the VFs reach the Active.Available state. See Section 2.4 for details.

3.3.9. First VF Offset (14h)

First VF Offset defines the requestor ID (RID) offset of the first VF that is associated with the PF that contains this capability structure. The first VFs 16-bit RID is calculated by adding the contents of this field to the RID of the PF containing this field ignoring any carry, using unsigned, 16-bit arithmetic.

No VF may be located on a bus number that is numerically smaller than its associated PF.

This field may change value when the VF ARI Enable value changes.

3.3.10. VF Stride (16h)

VF Stride must define the Requestor ID (RID) offset from one VF to the next one for all VFs associated with the PF that contains this capability structure. The next VFs 16-bit RID is calculated by adding the contents of this field to the RID of the current VF, ignoring any carry, using unsigned 16-bit arithmetic.

This field may change value when the VF ARI Enable value changes.

Note: VF Stride must not be zero.

3.3.11. Supported Page Sizes (18h)

This field must define the supported page sizes. This PF supports a page size of 2^(n+12) if bit n is Set. For example, if bit 0 is Set, the PF supports 4k byte page sizes. PFs are required to support 4k, 8k, 64k, 256k, 1M and 4M page sizes. All other page sizes are optional.

A page size describes the minimum alignment requirements for VF BAR resources as described in "System Page Size".

3.3.12. System Page Size (1Ch)

This field must define the page size the system will use to map the VFs' memory addresses. Software must set the value of the System Page Size to one of the page sizes set in the Supported Page Sizes field. As with Supported Page Sizes, if bit n is Set in System Page Size, the VFs associated with this PF are required to support a page size of 2^(n+12). For example, if bit 1 is Set, the system is using an 8k byte page size. The results are undefined if more than one bit is set in System Page Size. The results are undefined if a bit is Set in System Page Size that is not Set in Supported Page Sizes.

When System Page Size is set, the VF associated with this PF is required to align all BAR resources on a System Page Size boundary. Each VF BAR*n* or VF BAR*n* pair (described below) shall be aligned on a System Page Size boundary. Each VF BAR*n* or VF BAR*n* pair defining a non-zero address space shall be sized to consume an integer multiple of System Page Size bytes. All data structures requiring page size alignment within a VF shall be aligned on a System Page Size boundary.

VF Enable must be zero when System Page Size is set. The results are undefined if System Page Size is set when VF Enable is Set.

After a conventional reset, this field shall contain the value 0000 0000 0000 0001h.

3.3.13. VF BAR0, VF BAR1, ... VF BAR5 (20h .. 34h)

These fields must define the VFs Base Address Registers (BAR). These fields behave as normal PCI BARs, as described in the PCI Local Bus Specification 3.0. They can be sized by writing all 1's and reading back the contents of the BARs as described in the PCI Local Bus base specification, complying with the low order bits that define the BAR type fields. The amount of address space decoded by each BAR shall be an integral multiple of System Page Size.

Each VF BAR*n* register, when "sized" by writing 1's and reading back the contents, describes the amount of address space consumed and alignment required by a single virtual function, per BAR. When written with an actual address value, and VF Enable and VF MSE are Set, the BAR maps NumVFs BARs. In other words, the base address is address of the first VF BAR*n* associated with this PF and all subsequent VF BAR*n* address ranges follow as described below.

The alignment requirement and size read is for a single VF, but when VF Enable is set and VF MSE is set, the BAR contains the base address for all (NumVFs) VF BAR*n*.

The algorithm to determine the amount of address space mapped by a VF BAR*n* differs from the base specification as follows:

1. After reading the low order bits to determine if the BAR is a 32-bit BAR or 64-bit BAR pair, determine the size and alignment requirements by writing all 1s to VF BAR*n* (or VF BAR*n* and VF BAR*n+1* for a 64-bit BAR pair) and reading back the contents of the BAR or BAR pair. Convert the bit mask returned by the read(s) to a size and alignment value as described in the PCI Local Bus specification. This value is the size and alignment for a single VF.
2. Multiply the value from step1 by the value set in NumVFs to determine the total amount of space the BAR or BAR pair will map after VF Enable and VF MSE are Set.

For each VF BAR*n* field, *n* corresponds to one of the VFs BAR registers. The following table shows the relationship between *n* and a function's BAR registers.

n	Base Address Register offset in type 0 header
0	10h
1	14h
2	18h
3	1Ch
4	20h
5	24h

3.3.14. VF Migration State Array Offset (38h)

This field must contain a PF BAR relative pointer to the VF Migration State Array. The contents of the VF Migration State Array are defined below.

Table 3-5: VF Migration State Array Offset

Bit Location	Register Description	Attributes
31..3	**VF Migration State Offset** – Used as an offset from the address contained by one of the function's Base Address registers to point to the base of the VF Migration State Array. The lower 3 MVF Migration State BIR bits are masked off (set to zero) by software to form a 32-bit QWORD-aligned offset.	RO
2..0	**VF Migration State BIR** – Indicates which one of a function's Base Address registers, located beginning at 10h in Configuration Space, is used to map the function's VF Migration State Array into Memory Space. **BIR Value Base Address register** 0 BAR0 10h 1 BAR1 14h 2 BAR2 18h 3 BAR3 1Ch 4 BAR4 20h 5 BAR5 24h 6 Reserved 7 Reserved For a 64-bit Base Address register, the VF Migration State BIR indicates the lower DWORD.	RO

If a BAR that maps address space for the VF Migration State Array also maps other usable address space not associated with VF Migration, locations used in the other address space must not share any naturally aligned 8K byte address range with one where the VF Migration State Array resides.

This field is RO zero if VF Migration Capable is Clear.

3.3.14.1. VF Migration State Array

The VF Migration State Array is located using the VF Migration State Array Offset field of the SR-IOV capability block.

The VF Migration State Array a VF Migration State Entry for each VF. The total size of the VF Size array is NumVFs bytes. Table 3-6 defines the layout of a VF Migration State Array entry.

Table 3-6: VF Migration State Entry

Bit Location	Register Description	Attributes
1..0	**VF Migration State** – State of the associated VF	RW
7..1	**Reserved** – These fields are currently reserved	ReservedP

VF Migration State contains the following values.

Table 3-7: VF Migration State Descriptions

VF State	VF Exists	Description
00b	No	**Inactive.Unavailable** – VF does not exist to SR nor is it being migrated in or out.
01b	No	**Inactive.MigrateIn** – VF is available for use by SR. VF does not yet exist but will once the VF is Activated.
10b	Yes	**Active.MigrateOut** – SR has been requested to relinquish use of the VF.
11b	Yes	**Active.Available** – Fully functional. Could be assigned to an SI.

Software initiates a state transition by writing a new state value to the entry.

Valid state transitions are listed in Table 3-8 and Table 3-9 and shown in Figure 2-3. Only changes in Table 3-8 can be requested SR. Changes in Table 3-9 are initiated by MR-PCIM using mechanisms described in the MR-IOV Specification and have SR visible effects described here. Any transition issued by SR-PCIM and not listed in Table 3-8 is ignored and does not change the VF State.

Table 3-8: SR-PCIM Initiated VF Migration State Transitions

Current State	New State	Change Initiated By	SR Visible Effects of Change
Inactive.MigrateIn	Active.Available	SR-PCIM	**VF Activate** VF now exists
Active.Available	Inactive.MigrateIn	SR-PCIM	**VF Deactivate** VF no longer exists.
Active.MigrateOut	Inactive.Unavailable	SR-PCIM	**VF Migrate Out Complete** VF no longer exists.

Table 3-9: MR-PCIM Initiated VF Migration State Transitions

Current State	New State	Change Initiated By	SR Visible Effects of Change
Active.Available	Active.MigrateOut	MR-PCIM	**VF Migrate Out Request** VF continues to exist. Sets VF Migration Event Pending.
Inactive.Unavailable	Inactive.MigrateIn	MR-PCIM	**VF Migrate In Request** VF remains non-existent. Sets VF Migration Event Pending.
Inactive.MigrateIn	Inactive.Unavailable	MR-PCIM	**VF Migrate In Retract** VF remains non-existent. Sets VF Migration Event Pending.
Active.MigrateOut	Active.Available	MR-PCIM	**VF Migrate Out Retract** VF continues to exist. Sets VF Migration Event Pending.
Active.MigrateOut	Inactive.Unavailable	MR-PCIM	**VF Migrate Out Forced Complete** VF no longer exists. Sets VF Migration Event Pending.
Active.Available	Inactive.Unavailable	MR-PCIM	**VF Migrate Out Forced Complete** VF no longer exists. Sets VF Migration Event Pending. Note: The VF transitioned through Active.MigrateOut but SR-PCIM may not have seen the transition.

3.4. PF/VF Configuration Space Header

This section defines the requirements on PF and VF configuration space fields.

Devices compliant with this Specification must comply with PCI Express Base 1.1 or later. The register definitions listed throughout this document establish the mapping between existing PCI-SIG Specifications and the PF/VF definitions introduced in this Specification.

Table 3-10: Register Attributes (Modified from PCIe Base 1.0 draft)

Register Attribute	Description
LB 3.0	Attribute is same as specified in PCI Local Bus Specification 3.0.
Base 1.1	Attribute is same as specified in PCI Express Base Specification, Revision 1.1
Base 2.0	Attribute is same as specified in PCI Express Base Specification, Revision 2.0.

Register Attribute	Description
HwInit	Hardware Initialized: Register bits are initialized by firmware or hardware mechanisms such as pin strapping or serial EEPROM. Bits are read-only after initialization and can only be reset (for write-once by firmware) with "Power Good Reset" (see Section **Error! Reference source not found.**).
RO	Read-only register: Register bits are read-only and cannot be altered by software. Register bits may be initialized by hardware mechanisms such as pin strapping or serial EEPROM.
RW	Read-Write register: Register bits are read-write and may be either set or cleared by software to the desired state.
RW1C	Read-only status, Write-1-to-clear status register: Register bits indicate status when read, a set bit indicating a status event may be cleared by writing a 1. Writing a 0 to RW1C bits has no effect.
ROS	Sticky - Read-only register: Registers are read-only and cannot be altered by software. Registers are not initialized or modified by reset. Devices that consume AUX power must preserve sticky register values when AUX power consumption (either via AUX power or PME Enable) is enabled.
RWS	Sticky - Read-Write register: Registers are read-write and may be either set or cleared by software to the desired state. Bits are not initialized or modified by reset. Devices that consume AUX power must preserve sticky register values when AUX power consumption (either via AUX power or PME Enable) is enabled.
RW1CS	Sticky - Read-only status, Write-1-to-clear status register: Registers indicate status when read, a set bit indicating a status event may be cleared by writing a 1. Writing a 0 to RW1CS bits has no effect. Bits are not initialized or modified by reset. Devices that consume AUX power must preserve sticky register values when AUX power consumption (either via AUX power or PME Enable) is enabled.
RsvdP	Reserved and Preserved: Reserved for future RW implementations; software must preserve value read for writes to bits.
RsvdZ	Reserved and Zero: Reserved for future RW1C implementations; software must use 0 for writes to bits.

3.4.1. Type 0 Configuration Space Header

Figure 3-2 details allocation for register fields of PCI Express Type 0 Configuration Space Header for VFs.

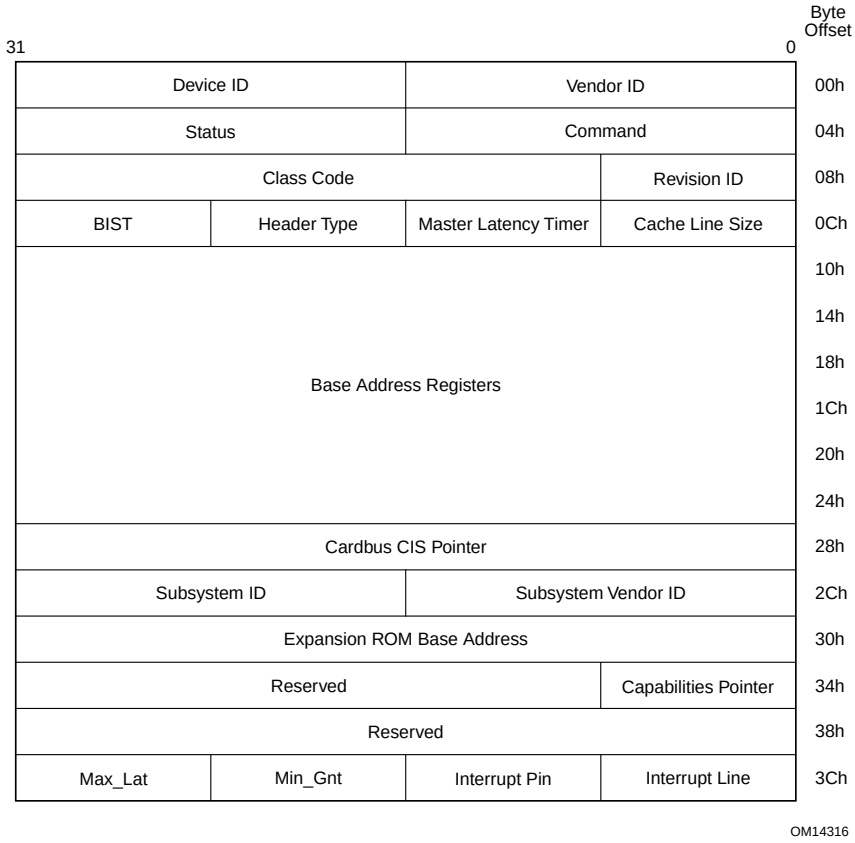


Figure 3-2: Type 0 Configuration Space Header

The PCI Express-specific interpretation of registers specific to PCI Express Type 0 Configuration Space Header is defined in this section.

3.4.1.1. Vendor ID Register (Offset 00h)

This Read Only field identifies the manufacturer of the Device. The field in the PF and associated VFs must return the same value when read.

3.4.1.2. Device ID Register (Offset 02h)

This Read Only field identifies the particular device. The Device may report a different value in the PF and the VF; however all VFs associated with a given PF must report the same value.

3.4.1.3. Command Register (Offset 04h)

Table 3-11 establishes the mapping between PCI Express Base 2.0 and PF/VF definitions in this document for configuration space Command register.

Table 3-11: Command Register

Bit Location	PF and VF Register Differences from Base 2.0	PF Attributes	VF Attributes
0	**I/O Space Enable –** Does not apply to VFs. Must be hardwired to 0b for VFs.	Base 1.1	RO
1	**Memory Space Enable –** Does not apply to VFs. Must be hardwired to 0b for VFs. VF Memory Space is controlled by the VF MSE bit in the VF Control Register.	Base 1.1	RO
2	**Bus Master Enable** – Disabling this bit prevents the associated PF or VF from issuing any Memory or I/O Requests. Note that as MSI/MSI-X interrupt Messages are in-band memory writes, disabling the bus master enable bit disables MSI/MSI-X interrupt Messages as well. Requests other than Memory or I/O Requests are not controlled by this bit. Default value of this field is 0b. This bit is hardwired to 0b if a device does not generate Memory or I/O Requests. Note: The state of active transactions is not specified when this bit is disabled after being enabled. The Device can choose how it behaves when this condition occurs. Software cannot count on the Device retaining state and resuming without loss of data when the bit is re-enabled. Transactions for a VF that has its Bus Master Enable set must not be blocked by transactions for VFs that have their Bus Master Enable cleared.	Base 1.1	Base 1.1
3	**Special Cycle Enable –** Does not apply to PCI Express and must be hardwired to 0b	Base 1.1	Base 1.1
4	**Memory Write and Invalidate –** Does not apply to PCI Express and must be hardwired to 0b	Base 1.1	Base 1.1
5	**VGA Palette Snoop –** Does not apply to PCI Express and must be hardwired to 0b	Base 1.1	Base 1.1
6	**Parity Error Enable** – Does not apply to VFs. Must be hardwired to 0b for VFs. See <Section X> for details on error reporting	Base 1.1	RO
7	**IDSEL Stepping/Wait Cycle Control –** Does not apply to PCI Express and must be hardwired to 0b	Base 1.1	Base 1.1

Bit Location	PF and VF Register Differences from Base 2.0	PF Attributes	VF Attributes
8	**SERR Enable –** Does not apply to VFs. Must be hardwired to 0b for VFs. See <Section X> for details on error reporting	Base 1.1	RO
9	**Fast Back-to-Back Transactions Enable–** Does not apply to PCI Express and must be hardwired to 0b	Base 1.1	Base 1.1
10	**Interrupt Disable –** Does not apply to VFs. Must be hardwired to 0b for VFs.	Base 1.1	RO

3.4.1.4. Status Register (Offset 06h)

Table 3-12 establishes the mapping between PCI Express Base 1.1 and PF/VF definitions in this document for configuration space Status register.

Table 3-12: Status Register

Bit Location	PF and VF Register Differences from Base 1.1	PF Attributes	VF Attributes
3	**Interrupt Status –** Does not apply to VFs. Must be hardwired to 0b for VFs.	Base 1.1	RO
4	**Capabilities List –** Must be hardwired to 1 for PFs and VFs.	Base 1.1	Base 1.1
5	**66 MHz Capable –** Does not apply to PCI Express and must be hardwired to 0b	Base 1.1	Base 1.1
7	**Fast Back-to-Back Transactions Capable –** Does not apply to PCI Express and must be hardwired to 0b	Base 1.1	Base 1.1
8	**Master Data Parity Error –** See <Section X>	Base 1.1	Base 1.1
10:9	**DEVSEL Timing –** Does not apply to PCI Express and must be hardwired to 0b	Base 1.1	Base 1.1
11	**Signaled Target Abort –** See <Section X>	Base 1.1	Base 1.1
12	**Received Target Abort –** See <Section X>	Base 1.1	Base 1.1
13	**Received Master Abort –** See <Section X>	Base 1.1	Base 1.1
14	**Signaled System Error –** See <Section X>	Base 1.1	Base 1.1
15	**Detected Parity Error –** See <Section X>	Base 1.1	Base 1.1

3.4.1.5. Revision ID Register (Offset 08h)

This register specifies a device specific revision identifier. This field should be viewed as a vendor defined extension to the *Device ID* and follows the same rules regarding PFs and VFs.
The value reported in the VF may be different than the value reported in the PF; however, all VFs associated with a given PF must report the same value.

3.4.1.6. Class Code (Offset 09h)

The Class Code register is read-only and is used to identify the generic function of the device and, in some cases, a specific register level programming interface. The field in the PF and associated VFs must return the same value when read.

3.4.1.7. Cache Line Size Register (Offset 0Ch)

This field is implemented by PCI Express devices as a read-write field for legacy compatibility purposes but has no effect on any PCI Express device behavior. Physical Functions continue to implement this field as RW. For Virtual Functions this field is RO and must be hardwired 00h

3.4.1.8. Latency Timer Register (Offset 0Dh)

This field does not apply to PCI Express. This register must be hardwired to 00h.

3.4.1.9. Header Type (Offset 0Eh)

This byte identifies the layout of the second part of the predefined header (beginning at byte 10h in Configuration Space) and also whether or not the device contains multiple Functions. Bit 7 in this register is used to identify a multi-function device. For an SR-IOV device, bit 7 is only Set if there are multiple PFs or Functions. VFs do not affect the value of bit 7. Bits 6 through 0 identify the layout of the second part of the predefined header. For VFs, this field must be 00h.

3.4.1.10. BIST (Offset 0Fh)

This optional register is used for control and status of BIST. Devices that do not support BIST must always return a value of 0 (i.e., treat it as a reserved register).

PFs and VFs shall not support BIST and must define this field as RO Zero.

If a multi-function Device supports both IOV and non-IOV Functions, BIST operations on a non-IOV Function must not impact IOV Functions, i.e. BIST operations must not impact the resources associated with the IOV Functions.

3.4.1.11. Base Address Registers (Offset 10h, 14h, ... 24h)

For VFs, the values in these registers are Read Only Zero.

Note: See also sections 2.1.2.1 and 3.3.13.

3.4.1.12. Cardbus CIS Pointer (Offset 28h)

For VFs, this register is not used and shall return Zero.

3.4.1.13. Subsystem Vendor ID (Offset 2Ch)

This Read Only field identifies the manufacturer of the subsystem. The field in the PF and associated VFs must return the same value when read.

3.4.1.14. Subsystem Device ID (Offset 2Eh)

This Read Only field identifies the particular subsystem. The Device may have a different value in the PF and the VF; however all VFs associated with a given PF must have the same value.

3.4.1.15. Expansion ROM BAR (Offset 30h)

The Expansion ROM BAR may be implemented in PFs. The expansion ROM BAR is R in VFs. The VI may choose to provide access to the PF ROM BAR for VFs via emulation if such access is required.

ROM BAR shared decoding, defined in the PCI Local Bus Specification 3.0, section 6.2.5.2 and similarly in other versions of the PCI Local Bus Specification is not permitted in any Device compliant with this specification.

3.4.1.16. Capabilities Pointer (Offset 34h)

No differences from PCIe specification.

3.4.1.17. Interrupt Line Register (Offset 3Ch)

This field does not apply to VFs. This field must be hardwired to 0.

3.4.1.18. Interrupt Pin Register (Offset 3Dh)

This field does not apply to VFs. This field must be hardwired to 0.

3.4.1.19. Min_Gnt / Max_Lat Registers (Offset 3Eh/3Fh)

These registers do not apply to PCI Express. They must be RO and hardwired to 0.

3.5. PCI Express Capability

PCI Express defines a capability structure in PCI 2.3 compatible configuration space (first 256 bytes) as shown in Figure 3-2 for identification of a PCI Express device and indicates support for new PCI Express features. The PCI Express Capability Structure is required for PCI Express devices. The capability structure is a mechanism for enabling PCI software transparent features requiring support on legacy operating systems. In addition to identifying a PCI Express device, the PCI Express

Capability Structure is used to provide access to PCI Express specific Control/Status registers and related Power Management enhancements.

Figure 3-3 details allocation of register fields in the PCI Express Capability Structure. As described below, the PCI Express Capabilities, Device Capabilities, Device Status/Control, Link Capabilities and Link Status/Control registers are required for PFs in PCI Express SR-IOV devices. As described below, VFs must implement portions of the PCI Express Capabilities, Device Capabilities, and Device Status/Control registers. Endpoints are not required to implement registers other than those listed above and terminate the capability structure.

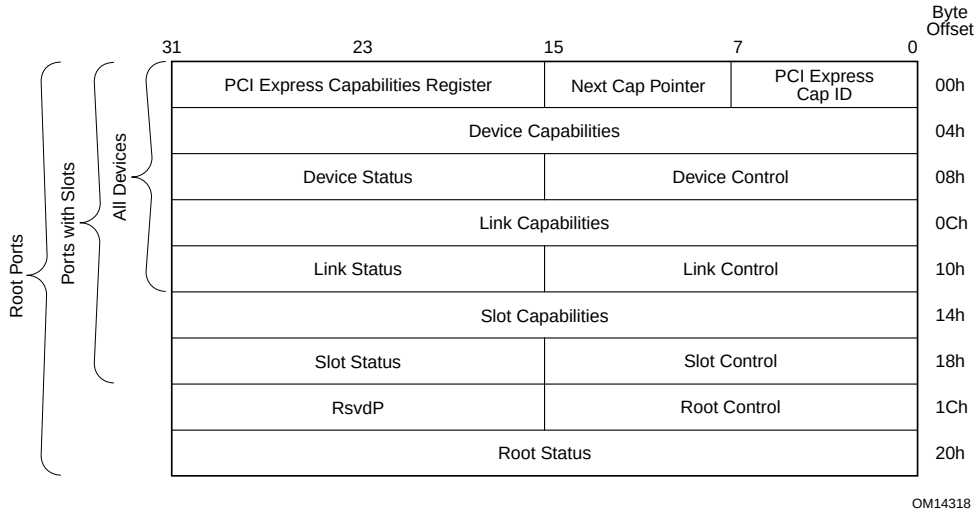


Figure 3-3: PCI Express Capability Structure

3.5.1. PCI Express Capability List Register (Offset 00h)

The PCI Express Capability List register enumerates the PCI Express Capability Structure in the PCI 2.3 configuration space capability list. Figure 3-4 details allocation of register fields in the PCI Express Capability List register; Table 3-13 provides the respective bit definitions.

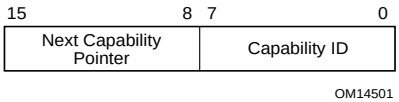


Figure 3-4: PCI Express Capability List Register

Table 3-13: PCI Express Capability List Register

Bit Location	PF and VF Register Differences from Base 1.1	PF Attributes	VF Attributes
7:0	**Capability ID**	Base 1.1	Base 1.1
15:8	**Next Capability Pointer**	Base 1.1	Base 1.1

3.5.2. PCI Express Capabilities Register (Offset 02h)

The PCI Express Capabilities register identifies PCI Express device type and associated capabilities. Figure 3-5 details allocation of register fields in the PCI Express Capabilities register; Table 3-14 provides the respective bit definitions.

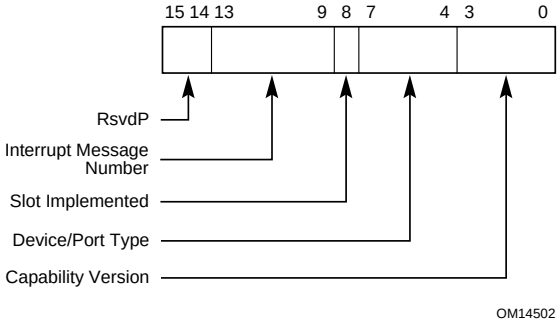


Figure 3-5: PCI Express Capabilities Register

Table 3-14: PCI Express Capabilities Register

Bit Location	PF and VF Register Differences from Base 1.1	PF Attributes	VF Attributes
3:0	**Capability Version**	Base 1.1	Base 1.1
7:4	**Device/Port Type** – SR-IOV capable functions must indicate a Device Type of 0000b (PCI Express Endpoint) Support for 1001b (Root Complex Integrated Endpoint) will be added after Version 0.7 of this specification.	Base 1.1	Base 1.1
8	**Slot Implemented** – Does not apply to PFs or VFs. Must be hardwired to 0b.	Base 1.1	Base 1.1
13:9	**Interrupt Message Number**	Base 1.1	Base 1.1
14	Must be hardwired to 0b. Note: This field supported TCS Routing. It was added in an ECN to the Base 1.1 specification and was removed in the Base 2.0 specification.	RO	RO

3.5.3. Device Capabilities Register (Offset 04h)

The Device Capabilities register identifies PCI Express device specific capabilities. Figure 3-6 details allocation of register fields in the Device Capabilities register; Table 3-15 provides the respective bit definitions.

Deleted: Figure 3-6

Deleted: Table 3-15

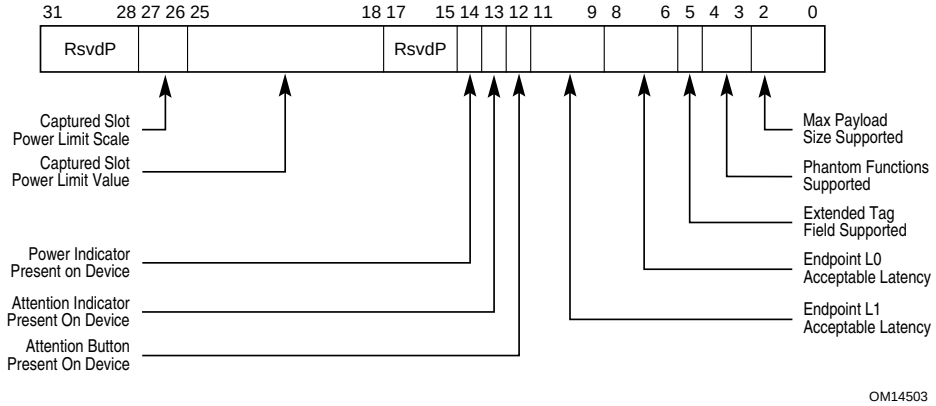


Figure 3-6: Device Capabilities Register

Table 3-15: Device Capabilities Register

Bit Location	PF and VF Register Differences from Base 1.1	PF Attributes	VF Attributes
2:0	**Max_Payload_Size Supported**	Base 1.1	Base 1.1
4:3	**Phantom Functions Supported** – When the SR-IOV Capability is enabled, use of Phantom Function numbers by this PF and associated VFs is not permitted and this register must return 00b when read.	Base 1.1	RO
5	**Extended Tag Field Supported**	Base 1.1	Base 1.1
8:6	**Endpoint L0s Acceptable Latency**	Base 1.1	Base 1.1
11:9	**Endpoint L1 Acceptable Latency**	Base 1.1	Base 1.1
12	**Undefined** – The value read from this bit is undefined in Base 1.1 (was previously Attention Button Present).	Base 1.1	Base 1.1
13	**Undefined** – The value read from this bit is undefined in Base 1.1 (was previously Attention Indicator Present).	Base 1.1	Base 1.1
14	**Undefined** – The value read from this bit is undefined in Base 1.1 (was previously Power Indicator Present).	Base 1.1	Base 1.1
15	**Role-Based Error Reporting** This bit must be set for devices that conform to this specification.	Base 1.1	Base 1.1
25:18	**Captured Slot Power Limit Value –** This field is hardwired 0 in the VFs	Base 1.1	RO
27:26	**Captured Slot Power Limit Scale -** This field is hardwired 0 in the VFs	Base 1.1	RO

Bit Location	PF and VF Register Differences from Base 1.1	PF Attributes	VF Attributes
28	**Function Level Reset Capability –** Required for SR-IOV devices (PFs and VFs). Must be hardwired to 1. Note: This field added as an ECN to the Base 1.1 specification and is included in the Base 2.0 specification.	Base 1.1 + ECN	Base 1.1 + ECN

3.5.4. Device Control Register (Offset 08h)

The Device Control register controls PCI Express device specific parameters. Figure 3-7 details allocation of register fields in the Device Control register; Table 3-16 provides the respective bit definitions.

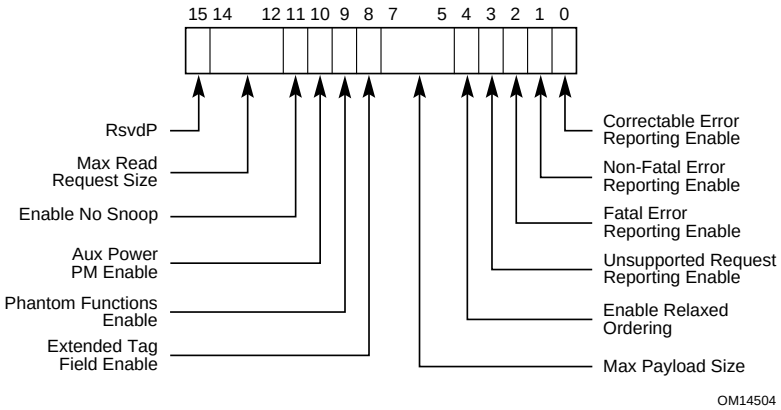


Figure 3-7: Device Control Register

Table 3-16: Device Control Register

Bit Location	PF and VF Register Differences from Base 1.1	PF Attributes	VF Attributes
0	**Correctable Error Reporting Enable -** Does not apply to VFs. Must be hardwired to 0b for VFs. See <Section X> for details on error reporting	Base 1.1	RO
1	**Non-Fatal Error Reporting Enable-** Does not apply to VFs. Must be hardwired to 0b for VFs. See <Section X> for details on error reporting	Base 1.1	RO
2	**Fatal Error Reporting Enable -** Does not apply to VFs. Must be hardwired to 0b for VFs. See <Section X> for details on error reporting	Base 1.1	RO

Bit Location	PF and VF Register Differences from Base 1.1	PF Attributes	VF Attributes
3	**Unsupported Request Reporting Enable -** Does not apply to VFs. Must be hardwired to 0b for VFs. See <Section X> for details on error reporting	Base 1.1	RO
4	**Enable Relaxed Ordering –** Must be hardwired to 0b for VFs.	Base 1.1	RO
7:5	**Max_Payload_Size –** PF setting applies to all associated VFs as well. Must be hardwired to 0b for VFs	Base 1.1	RO
8	**Extended Tag Field Enable –** PF setting applies to all associated VFs as well. Must be hardwired to 0b for VFs	Base 1.1	RO
9	**Phantom Functions Enable** – If SR-IOV is enabled, this bit is hardwired to 0.	Base 1.1	RO
10	**Auxiliary (AUX) Power PM Enable** Must be hardwired to 0b for VFs	Base 1.1	RO
11	**Enable No Snoop –** PF setting applies to all associated VFs as well. Must be hardwired to 0b for VFs	Base 1.1	RO
14:12	**Max_Read_Request_Size** – PF setting applies to all associated VFs as well. Must be hardwired to 0b for VFs	Base 1.1	RO
15	**Initiate Function Level Reset –** Required for SR-IOV devices (PFs and VFs). Note: This field added as an ECN to the Base 1.1 specification and is included in the Base 2.0 specification.	Base 1.1 + ECN	Base 1.1 + ECN

3.5.5. Device Status Register (Offset 0Ah)

The Device Status register provides information about PCI Express device specific parameters. Figure 3-8 details allocation of register fields in the Device Status register; Table 3-17 provides the respective bit definitions.

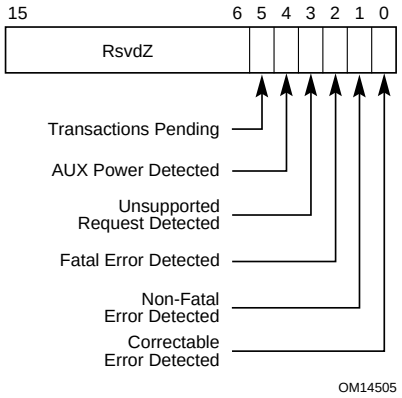


Figure 3-8: Device Status Register

Table 3-17: Device Status Register

Bit Location	PF and VF Register Differences from Base 1.1	PF Attributes	VF Attributes
0	**Correctable Error Detected**	Base 1.1	Base 1.1
1	**Non-Fatal Error Detected**	Base 1.1	Base 1.1
2	**Fatal Error Detected**	Base 1.1	Base 1.1
3	**Unsupported Request Detected**	Base 1.1	Base 1.1
4	**AUX Power Detected** Must be hardwired to 0b for VFs	Base 1.1	RO
5	**Transactions Pending** – When set indicates that a particular function (PF or VF) has issued Non-Posted Requests which have not been completed. A function reports this bit cleared only when all Completions for any outstanding Non-Posted Requests have been received.	Base 1.1	Base 1.1

3.5.6. Link Capabilities Register (Offset 0Ch)

The Link Capabilities register identifies PCI Express Link specific capabilities. Figure 3-9 details allocation of register fields in the Link Capabilities register; Table 3-18 provides the respective bit definitions.

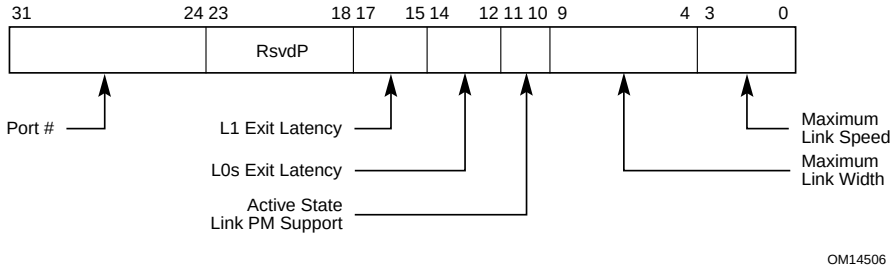


Figure 3-9: Link Capabilities Register

Table 3-18: Link Capabilities Register

Bit Location	PF and VF Register Differences from Base 1.1	PF Attributes	VF Attributes
3:0	**Supported Link Speeds**	Base 1.1	Base 1.1
9:4	**Maximum Link Width**	Base 1.1	Base 1.1
11:10	**Active State Power Management (ASPM) Support**	Base 1.1	Base 1.1
14:12	**L0s Exit Latency**	Base 1.1	Base 1.1
17:15	**L1 Exit Latency**	Base 1.1	Base 1.1
18	**Clock Power Management**	Base 1.1	Base 1.1
19	**Surprise Down Error Reporting Capable –** For Upstream Ports, this bit must be hardwired to 0b.	Base 1.1	Base 1.1
20	**Data Link Layer Link Active Reporting Capable –** For Upstream Ports, this bit must be hardwired to 0b.	Base 1.1	Base 1.1
21	**Link Bandwidth Notification Capability** – Not applicable to Endpoint devices. Must be hardwired to 0b.	Base 2.0	Base 2.0
31:24	**Port Number**	Base 1.1	Base 1.1

If L1 state is not supported for Active State Link PM (as reported in the Active State Link PM Support field) then the L1 Exit latency field is ignored.

3.5.7. Link Control Register (Offset 10h)

The Link Control register controls PCI Express Link specific parameters. Figure 3-10 details allocation of register fields in the Link Control register; Table 3-19 provides the respective bit definitions.

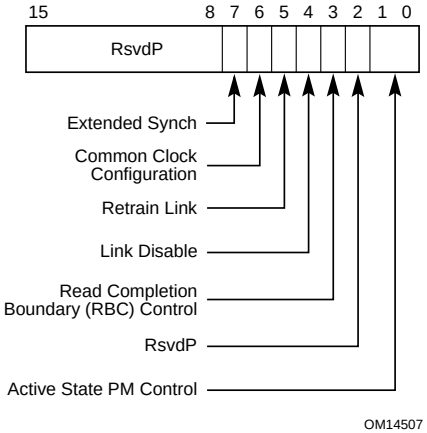


Figure 3-10: Link Control Register

Table 3-19: Link Control Register

Bit Location	PF and VF Register Differences from Base 1.1	PF Attributes	VF Attributes
1:0	**Active State Power Management (ASPM) Control** Must be hardwired to 0b for VFs	Base 1.1	RO
3	**Read Completion Boundary (RCB)** Must be hardwired to 0b for VFs	Base 1.1	RO
4	**Link Disable** – Reserved for Endpoint devices. Must be hardwired to 0b.	Base 1.1	Base 1.1
5	**Retrain Link** – Reserved for Endpoint devices. Must be hardwired to 0b.	Base 1.1	Base 1.1
6	**Common Clock Configuration** Must be hardwired to 0b for VFs	Base 1.1	RO
7	**Extended Synch** Must be hardwired to 0b for VFs	Base 1.1	RO
8	**Enable Clock Power Management** Must be hardwired to 0b for VFs	Base 1.1	RO
9	**Hardware Autonomous Width Disable** Must be hardwired to 0b for VFs	Base 2.0	RO
10	**Link Bandwidth Management Interrupt Enable** – Not applicable to Endpoint devices. Must be hardwired to 0b.	Base 2.0	Base 2.0
11	**Link Autonomous Bandwidth Interrupt Enable** – Not applicable to Endpoint devices. Must be hardwired to 0b.	Base 2.0	Base 2.0

3.5.8. Link Status Register (Offset 12h)

The Link Status register provides information about PCI Express Link specific parameters. Figure 3-11 details allocation of register fields in the Link Status register; Table 3-20 provides the respective bit definitions.

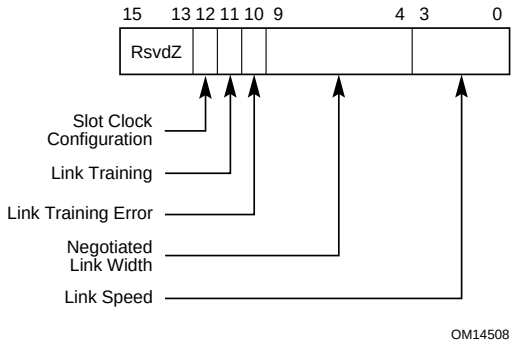


Figure 3-11: Link Status Register

Table 3-20: Link Status Register

Bit Location	PF and VF Register Differences from Base 1.1	PF Attributes	VF Attributes
3:0	**Current Link Speed** Must be hardwired to 0b for VFs	Base 1.1	RO
9:4	**Negotiated Link Width** Must be hardwired to 0b for VFs	Base 1.1	RO
10	**Undefined** – The value read from this bit is undefined in Base 1.1 (was previously Training Error).	Base 1.1	Base 1.1
11	**Link Training** – Reserved for Endpoint devices. Must be hardwired to 0b.	Base 1.1	Base 1.1
12	**Slot Clock Configuration** Must be hardwired to 0b for VFs	Base 1.1	RO
13	**Data Link Layer Link Active** Must be hardwired to 0b for VFs	Base 1.1	RO
14	**Link Bandwidth Management Status** – Reserved for Endpoint devices. Must be hardwired to 0b.	Base 2.0	Base 2.0
15	**Link Autonomous Bandwidth Status** – Reserved for Endpoint devices. Must be hardwired to 0b.	Base 2.0	Base 2.0

3.5.9. Device Capabilities 2 Register (Offset 24h)

Bit Location	PF and VF Register Differences from Base 1.1	PF Attributes	VF Attributes
3:0	**Completion Timeout Ranges Supported** Must be hardwired to 000b for VFs. Note: This field was added as an ECN to the Base 1.1 specification and is included in the Base 2.0 specification.	Base 1.1 + ECN	RO
4	**Completion Timeout Disable Supported** Must be hardwired to 0b for VFs. Note: This field was added as an ECN to the Base 1.1 specification and is included in the Base 2.0 specification.	Base 1.1 + ECN	RO

3.5.10. Device Control 2 Register (Offset 28h)

Bit Location	PF and VF Register Differences from Base 2.0	PF Attributes	VF Attributes
3:0	**Completion Timeout Value** Must be hardwired to 0000b for VFs Note: This field was added as an ECN to the Base 1.1 specification and is included in the Base 2.0 specification.	Base 1.1 + ECN	RO
4	**Completion Timeout Disable** Must be hardwired to 0b for VFs Note: This field was added as an ECN to the Base 1.1 specification and is included in the Base 2.0 specification.	Base 1.1 + ECN	RO

3.5.11. Device Status 2 Register (Offset 2Ah)

This section is a placeholder. There are no capabilities that require this register.

3.5.12. Link Capabilities 2 Register (Offset 2Ch)

This section is a placeholder. There are no capabilities that require this register.

3.5.13. Link Control 2 Register (Offset 30h)

Bit Location	PF and VF Register Differences from Base 2.0	PF Attributes	VF Attributes

Bit Location	PF and VF Register Differences from Base 2.0	PF Attributes	VF Attributes
3:0	**Target Link Speed** Must be hardwired to 0b for VFs	Base 2.0	RO
4	**Enter Compliance** Must be hardwired to 0b for VFs	Base 2.0	RO
5	**Hardware Autonomous Speed Disable** Must be hardwired to 0b for VFs	Base 2.0	RO
6	**Selectable De-emphasis** – Reserved for Endpoint devices. Must be hardwired to 0b.	Base 2.0	Base 2.0
9:7	**Transmit Margin** Must be hardwired to 0b for VFs	Base 2.0	RO
10	**Enter Modified Compliance** Must be hardwired to 0b for VFs	Base 2.0	RO

3.5.14. Link Status 2 Register (Offset 32h)

Bit Location	PF and VF Register Differences from Base 2.0	PF Attributes	VF Attributes
0	**Current De-emphasis Level** Must be hardwired to 0b for VFs	Base 2.0	RO

3.6. PCI Standard Capabilities

3.6.1. PCI Power Management Interface

This Capability is required for PFs and VFs. Additional requirements for VFs are described in Chapter 6.

3.6.2. AGP Capability

Not applicable to this specification.

3.6.3. VPD Capability

The VPD capability is optional (as with the base spec) in PFs. The VPD capability is not specified for VFs and may be implemented; however, it is a requirement that there can be no "data leakage" between VFs and/or PFs via the VPD capability if it's implemented in VFs.

3.6.4. Slot Identification

As defined by the base specifications. Not applicable to VFs.

3.6.5. MSI

Physical and virtual functions must either implement the MSI Capability or the MSI-X Capability or both MSI and MSI-X capabilities. See <Chapter X> for details on the requirements.

3.6.6. CompactPCI Hot Swap

Not applicable to this specification.

3.6.7. PCI-X

Not applicable to this specification.

3.6.8. "Reserved for AMD"

Not applicable to this specification.

3.6.9. Vendor Specific

Undefined by this specification (up to the vendor).

3.6.10. Debug Port

Undefined by this specification.

3.6.11. Compact PCI Central Resource Control

Not applicable to this specification.

3.6.12. PCI Hot Plug

As defined by the base spec. Not applicable to virtual functions.

3.6.13. PCI Bridge Subsystem ID

Not applicable to endpoints. As defined by the base spec.

3.6.14. AGP 8x

Not applicable to this specification

3.6.15. Secure Device (AMD)

Not applicable to this specification.

3.6.16. PCI Express

Described in section 3.5.

3.6.17. MSI-X

Physical and virtual functions must either implement the MSI Capability or the MSI-X Capability or both MSI and MSI-X capabilities. See <Chapter X> for details on the requirements.

3.6.18. Serial ATA Data/Index Configuration

Not applicable to this specification.

3.7. PCI Express Extended Capabilities

3.7.1. Advanced Error Reporting

Error requirements are defined in Chapter X.

3.7.2. Virtual Channel (02h)

Optional in physical functions. Not applicable to virtual functions.

3.7.3. Device Serial Number

This field is optional for PFs. This field is not implemented by VFs.

3.7.4. Power Budgeting

This field is optional for PFs. This field is not implemented by VFs.

3.7.5. Root Complex Link Declaration

Not applicable to this specification.

3.7.6. Root Complex Internal Link Control

Not applicable to this specification.

3.7.7. Root Complex Event Collector Endpoint Association

Not applicable to this specification.

3.7.8. MFVC

Optional in physical functions as defined by the base specification. Does not apply to VFs.

3.7.9. Virtual Channel (09h)

Optional in physical functions as defined by the base specification. Does not apply to VFs.

3.7.10. RCRB

Not applicable to this specification.

3.7.11. Vendor Specific

Not specified. (Up to the vendor).

3.7.12. CAC

The CAC capability is optional in physical functions and must not be implemented in virtual functions.

3.7.13. ACS

The ACS capability is optional and may be implemented in physical and virtual functions. Each function that implements the capability must implement the capability separately.

3.7.14. ARI

All Functions in SR-IOV Capable Devices (Devices that include at least one PF) must support the ARI capability.

For VFs, the *Next Function Number* field in the ARI Control register is Undefined.

Implementation Note: VFs are located using the SR-IOV Capability and are not included in the Device's ARI *Next Function Number* list. VFs use the ARI capability to export their Function Group Number as defined by the ACS capability.

3.7.15. ATS

ATS is optional in PFs and VFs.

VFs must implement the Enable (E) bit (bit 31 of the ATS Capability) if the PF implements the ATS Capability. When this field is set, the VF is enabled to invoke the ATS protocol and to cache address translations. Default value is 0b. RW

4

4. Error Handling

Native IOV devices utilize the same Error Reporting mechanism defined in the PCIe Base Specification Release 1.1.

The PCI Express Base Specification defines two error reporting paradigms: the baseline capability and the Advanced Error Reporting capability. The baseline error reporting capabilities are required of all PCI Express devices and define the minimum error reporting requirements. The Advanced Error Reporting capability is optional and is defined for more robust error reporting and is implemented with a specific PCI Express capability structure.

4.1. Baseline Error Reporting

All Native IOV devices must support the baseline error reporting capabilities, with some modifications to account for the goal of reduced cost and complexity of implementation.

When operating as an IOV endpoint device, there is a higher level Virtualizing Intermediary (VI) that is interposed between the SI and the VF. The VI intercepts configuration transactions and performs access control over configuration space. The VI is also the first responding agent to error messages and it may handle the issue itself and/or pass the error on to the SI. For the case of errors that are not function-specific the VI may pass the error on to multiple SIs.

The Error Reporting control registers are wholly owned by the VI and replicating these controls in all Virtual Functions adds unnecessary implementation cost to the endpoint device. These control bits are only present in the PF and all VFs will use the error reporting control bits in the corresponding PF when making decisions on generating error messages.

These following error control bits are not replicated in the VF and return the state of the PF when read:

- SERR# Enable
- Parity Error Response
- Correctable Reporting Enable
- Non-Fatal Reporting Enable
- Fatal Reporting Enable
- UR Reporting Enable

Each VF shall implement a mechanism to provide error status independent of any other Function. This is necessary to provide SI isolation for errors that are Function specific.

The following baseline error reporting status bits must be replicated in all VF:

- Master Data Parity Error

- Signaled Target Abort
- Received Target Abort
- Master Abort
- SERR# Asserted
- Detected Parity Error
- Correctable Error Detected
- Non-Fatal Error Detected
- Unsupported Request Detected

Each VF will use its own Requestor ID when signaling errors.

4.2. Advanced Error Reporting

The Advanced Error Reporting capability is optional. If AER is not implemented in the PF it must not be implemented in the associated VFs. If AER is implemented in the PF it is optional in the VFs.

These following error control bits are not replicated in the VF and return the state of the PF when read:

- Uncorrectable Error Mask Register
- Uncorrectable Error Severity Register
- Correctable Error Mask Register
- ECRC Generation Enable
- ECRC Check Enable

Note: These control bits are only present in the PF and all VFs will use the error reporting control bits in the corresponding PF when making decisions on generating error messages.

Each Virtual Function that implements the Advanced Error Reporting capability must maintain its own error reporting status for function-specific errors.

The following errors are considered function-specific errors:

- Poisoned TLP received
- Completion Timeout
- Completer Abort
- UR, when caused by a function that claims a TLP
- Unexpected Completion, when Requestor ID matches a Function in the Device
- ACS Violation

The other error types are defined by the PCI Express Base Specification as non function-specific. The non function-specific errors include All Physical Layer, All Data Link Layer, and the remaining Transaction Layer errors. For these errors the endpoint logs the error in the PF only and does not log it in the VFs.

The following PCI Express errors are not function-specific

- All Physical Layer errors

- All Data Link Layer errors
- ECRC Fail
- UR, when caused by no function claiming a TLP
- Receiver Overflow
- Flow Control Protocol Error
- Malformed TLP
- Unexpected Completion, when Requester ID doesn't match any Function in the Device

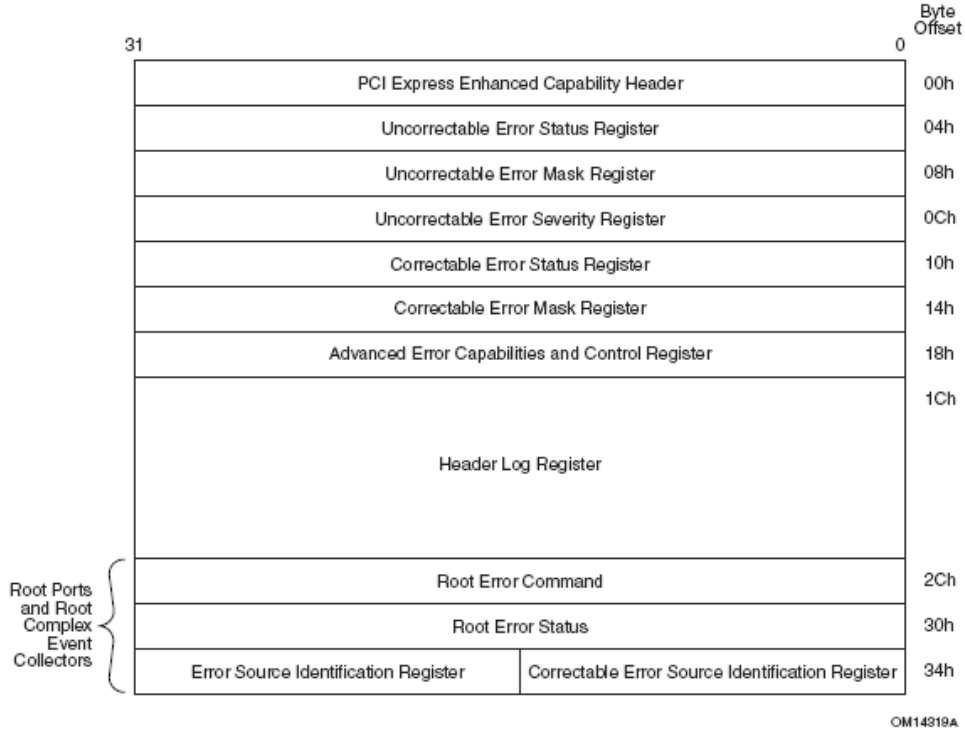
4.2.1. VF Header Log

A device that implements AER in the VFs may share Header Log Registers among VFs associated with a single PF. A shared header log must have storage for at least one header.

Header logging Registers for the PF is independent of its associated VFs and must be implemented with dedicated storage space.

When implementing a reduced set of Header Log Registers, a function may not have room to log a header associated with an error. In this case the function shall updated the Uncorrectable Error Status Register and Advanced Error Capabilities and Control register as required by the PCIe Base Specification; however, when the Header Log Register is read it shall return all '1's to indicate an overflow condition and no header was logged.

4.3. Advanced Error Reporting Capability



4.3.1. Advanced Error Reporting Enhanced Capability Header (Offset 00h)

This register contains the PCI Express Extended Capability ID, Capability Version, and Next Capability Offset. These fields have Base 1.1 attributes.

4.3.2. Uncorrectable Error Status Register (Offset 04h)

The Uncorrectable Error Status register indicates error detection status of individual errors. Errors that are defined as non-function specific are logged in the PF. Only function specific errors are logged in the VFs.

Table 4-1: PCI Express Capabilities Register

Bit Location	PF and VF Register Differences from Base 1.1	PF Attributes	VF Attributes
0	**Undefined**	Base 1.1	Base 1.1

Bit Location	PF and VF Register Differences from Base 1.1	PF Attributes	VF Attributes
4	**Data Link Protocol Error Status -** For VFs, this field is not applicable and must be hardwired to 0b.	Base 1.1	0b
5	**Surprise Down Error Status (Optional) -** For VFs, this field is not applicable and must be hardwired to 0b.	Base 1.1	0b
12	**Poisoned TLP Status**	Base 1.1	Base 1.1
14	**Completion Timeout Status**	Base 1.1	Base 1.1
15	**Completer Abort Status (Optional)**	Base 1.1	Base 1.1
16	**Unexpected Completion Status**	Base 1.1	Base 1.1
17	**Receiver Overflow Status -** For VFs, this field is not applicable and must be hardwired to 0b.	Base 1.1	0b
18	**Malformed TLP Status -** For VFs, this field is not applicable and must be hardwired to 0b.	Base 1.1	0b
19	**ECRC Error Status -** For VFs, this field is not applicable and must be hardwired to 0b.	Base 1.1	0b
20	**Unsupported Request Error Status**	Base 1.1	Base 1.1
21	**ACS Violation Status** Note: This field added as an ECN to the Base 1.1 specification and is included in the Base 2.0 specification.	Base 1.1 + ECN	Base 1.1 + ECN

4.3.3. Uncorrectable Error Mask Register (Offset 08h)

Uncorrectable Error Mask bits are only implemented in the PF. The VFs will use the error reporting control bits in the corresponding PF when making decisions on generating error messages.

This register is hardwired 0 in the VF.

4.3.4. Uncorrectable Error Severity Register (Offset 0Ch)

Uncorrectable Error Severity bits are only implemented in the PF. The VFs will use the error reporting control bits in the corresponding PF when making decisions on generating error messages.

This register is hardwired 0 in the VF.

4.3.5. Correctable Error Status Register (Offset 10h)

The Correctable Error Status register indicates error detection status of individual correctable errors. Errors that are defined as non-function specific are logged in the PF. Only function specific errors are logged in the VFs.

Bit Location	PF and VF Register Differences from Base 1.1	PF Attributes	VF Attributes
0	**Receiver Error Status –** For VFs, this field is not applicable and must be hardwired to 0b.	Base 1.1	RO
6	**Bad TLP Status –** For VFs, this field is not applicable and must be hardwired to 0b.	Base 1.1	RO
7	**Bad DLLP Status –** For VFs, this field is not applicable and must be hardwired to 0b.	Base 1.1	RO
8	**REPLAY_NUM Rollover Status –** For VFs, this field is not applicable and must be hardwired to 0b.	Base 1.1	RO
12	**Replay Timer Timeout Status –** For VFs, this field is not applicable and must be hardwired to 0b.	Base 1.1	RO
13	**Advisory Non-Fatal Error Status**	Base 1.1	Base 1.1

4.3.6. Correctable Error Mask Register (Offset 14h)

Correctable Error Mask bits are only implemented in the PF. The VFs will use the error reporting control bits in the corresponding PF when making decisions on generating error messages.

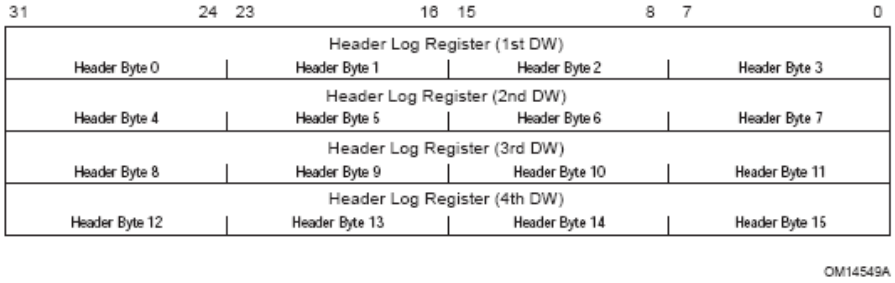
This register is hardwired 0 in the VF.

4.3.7. Advanced Error Capabilities and Control Register (Offset 18h)

Bit Location	PF and VF Register Differences from Base 1.1	PF Attributes	VF Attributes
4:0	**First Error Pointer**	Base 1.1	Base 1.1
5	**ECRC Generation Capable –** This field is hardwired 0b in the VF.	Base 1.1	RO
6	**ECRC Generation Enable –** This field is hardwired 0b in the VF.	Base 1.1	RO
7	**ECRC Check Capable –** This field is hardwired 0b in the VF.	Base 1.1	RO
8	**ECRC Check Enable –** This field is hardwired 0b in the VF.	Base 1.1	RO

4.3.8. Header Log Register (Offset 1Ch)

The Header Log register captures the header for the TLP corresponding to a detected error.



Bit Location	PF and VF Register Differences from Base 1.1	PF Attributes	VF Attributes
127:0	**Header of TLP associated with error**	Base 1.1	Base 1.1

4.3.9. Root Error Command Register (Offset 2Ch)

This register is not applicable to endpoint devices and is not covered by the SR-IOV specification

4.3.10. Root Error Status Register (Offset 30h)

This register is not applicable to endpoint devices and is not covered by the SR-IOV specification

4.3.11. Correctable Error Source Identification Register (Offset 34h)

This register is not applicable to endpoint devices and is not covered by the SR-IOV specification

4.3.12. Error Source Identification Register (Offset 36h)

This register is not applicable to endpoint devices and is not covered by the SR-IOV specification

5

5. Interrupts

Native IOV devices utilize the same Interrupt signaling mechanisms defined in the PCIe Base Specification Release 1.1.

An enhanced table-less type of MSI-X is currently being defined and will be proposed as an independent ECR to the PCIe Base Specification.

5.1. Requirements

5.1.1. PF Requirements

The following constraints apply to interrupt mechanisms on PF:

- Interrupt mechanisms on PF must be backward compatible with existing mechanisms in the PCI specification.
- INTx and associated CSRS must be supported.
- MSI or MSI-X or both must be supported.
- PF must support at least the MSI mechanisms supported by its associated VF.
- Any extensions to the MSI or MSI-X mechanisms must be backward compatible with the existing mechanisms.

5.1.2. VF Requirements

The following constraints apply to interrupt mechanisms on VF:

- INTx is not supported.

 Note: Although INTx is not directly supported by VF, a VI may be used to emulate INTx mechanisms.
- All VF have independent interrupt capabilities.
- All VF associated with a PF must:
 - Implement the same MSI type capabilities.
 - If MSI is implemented the corresponding capabilities must report the same value in the Multiple Message Capable field.

- If MSI-X is implemented the corresponding capabilities must report the same value in the Table Size field.

5.1.3. Common PF/VF Requirements

The following constraints apply to interrupt mechanisms on both VF and PF:

- The presence of an interrupt remapping facility in the RC must not be required for operation.
- The function must be able to operate without significant performance loss in a compatibility mode where only the vector field of the MSI is utilized to identify the interrupt.

 Note: Some architectures are limited to 256 vectors.
- Implementation of the mask and pending bits is required.
- Software must have the ability to manage the interrupts of a VF or PF independently of any other VF or PF.
- Each PF and VF must have its own unique interrupt capabilities.
- MSI messages must be tagged with the RID of the PF or VF that generated the interrupt.

5.2. Interrupt Mechanisms

There are three methods of signaling interrupts:

- INTx
- MSI
- MSI-X

PF must implement INTx. VF must not implement INTx. PFs and VFs must implement MSI, MSI-X or both. Each PF and VF must have its own unique interrupt capabilities.

5.2.1. MSI Interrupts

For PF or VF supporting MSI interrupts, the mechanism is identical to the currently defined mechanism in the existing PCI Specification with the follow exception:

- Implementation of the Mask and Pending bits is mandatory.

The VF MSI capabilities associated with each PF must report the same value in the Multiple Message Capable field.

If a VF implements a MSI capability, then all VF associated with the corresponding PF must implement a MSI capability. If a VF implements a MSI capability its associated PF must implement a MSI capability.

5.2.2. MSI-X Interrupts

For PF or VF supporting MSI-X interrupts the mechanism is identical to the currently defined mechanism in the existing PCI Specification.

The VF MSI-X capabilities associated with each PF must report the same value in the Table Size, Table Offset, Table BIR, PBA Offset, and PBA BIR fields.

If a VF implements a MSI-X capability then all VF associated with the corresponding PF must implement a MSI-X capability. If a VF implements a MSI-X capability its associated PF must implement a MSI-X capability.

5.2.3. Address range isolation

The current MSI-X specification requires the MSI-X Table and PBA regions to be aligned on separate 4 KB pages, with a strong recommendation that they be aligned on at least 8 KB pages. SR Endpoints have a System Page Size field that specifies the minimum separation that must exist between all VF memory structures. The System Page Size must be applied to the interrupt structures (VF's MSI-X Table and PBA) in addition to the above constraints.

5.2.4. MSI/MSI-X Capability field replication in VF and PF

Each PF and VF must fully implement either a MSI or a MSI-X capability or both. All fields must be implemented in each PF and VF.

5.2.5. MSI/MSI-X dependencies between PF and VF

All VFs associated with a particular PF must implement the same interrupt capabilities. A PF must, at a minimum, support the interrupt capabilities implemented by its associated VF. For example, if the VF associated with a PF implements a MSI-X capability the PF must implement a MSI-X capability. Additionally the PF may choose to implement a MSI capability.

6

6. Power Management

This chapter defines the PCI Express SR-IOV power management capabilities and protocols.

The Power Management Capability is required in both PFs and VFs as described in the PCIe Base Specification.

6.1. VF Device Power Management States

Support for D0 is required for VFs. Support for D1, D2 and D3 is optional for VFs.

If the VF supports only D0, the Power State field in the VF PMCSR is Read Only returning 00b.

Note: When the Device is placed into the $D3_{cold}$ state VF's are disabled. Therefore any VF specific context is lost. PME events can only be initiated by the PF.

6.2. Link Power Management State

VF Power State does not affect Link Power State.

Link Power State is managed through the PFs (including ASPM).

6.3. VF Power Management Capability

The following tables list the requirements for PF and VFs Power Management Capability.

Table 6-1: Power Management Capabilities - PMC

Bit Location	PF and VF Register Differences from Base 2.0	PF Attributes	VF Attributes
15:11	**PME_Support**	Base 1.1	Base 1.1
10	**D2_Support**	Base 1.1	Base 1.1
9	**D1_Support**	Base 1.1	Base 1.1
8:6	**Aux_Current**	Base 1.1	Base 1.1
5	**Device Specific Initialization (DSI)**	Base 1.1	Base 1.1
4	**Reserved**	Base 1.1	Base 1.1
3	**PME clock**	Base 1.1	Base 1.1
2:0	**Version**	Base 1.1	Base 1.1

Table 6-2: Power Management Control/Status – PMCSR

Bit Location	PF and VF Register Differences from Base 2.0	PF Attributes	VF Attributes
15	**PME_Status**	Base 1.1	Base 1.1
14:13	**Data_Scale**	Base 1.1	00b
12:9	**Data_Select**	Base 1.1	0000b
8	**PME_En**	Base 1.1	Base 1.1
7:4	**Reserved**	Base 1.1	Base 1.1
3	**No_Soft_Reset**	Base 1.1	Base 1.1
2	**Reserved**	Base 1.1	Base 1.1
1:0	**Power State**	Base 1.1	Base 1.1 or 00b

7

7. Acknowledgements

The following people were instrumental in the development of the SR-IOV Specification[1].

[ANY WHO FEEL THAT HAVE CONTRIBUTED TO THIS SPECIFICATION, SHOULD CONTACT MIKE DIRECTLY TO BE ADDED TO THIS SECTION. THIS IS STRICTLY AN OPT-IN SECTION PER THE PCI-SIG POLICIES]

[1] Company affiliation is at the time of specification contribution.